Development & Operations Security

Addendum to Information Security Policy

Polaris Group, LLC | Neucleos-1 Platform Services

Effective Date: December 2, 2024 | Version 1.1

1. Change Management & Release Controls

1.1 Source Control

All source code is managed in Git repositories hosted on GitHub with the following controls:

- **Version Control:** All code changes tracked with full commit history
- **Branch Protection:** Main branch protected; changes require pull requests
- **Code Review:** All changes reviewed before merge (self-review for solo operations with documented checklist)
- **Signed Commits:** GPG-signed commits for authenticity verification

1.2 Environment Separation

The platform maintains strict separation between development and production environments:

Environment	Identifier	Purpose
Development	Convex: canny-wildcat-911	Active development and testing
Production	Separate instances	Live customer-facing services

1.3 Deployment Process

Code deployment follows a defined process with documented guards:

1. **Development:** Changes developed and tested locally
2. **Commit:** Code committed to feature branch with descriptive message
3. **Review:** Pull request created; code reviewed against checklist
4. **Test:** Automated checks run (linting, type checking, build verification)
5. **Merge:** Approved changes merged to main branch
6. **Deploy:** Automatic deployment triggered via CI/CD pipeline

1.4 Deployment Guards

Critical deployment guards are documented and enforced:

- **Convex Deployment:** Development deployment only (npx convex dev --once); production deployment requires explicit approval
- **Schema Changes:** Database schema modifications require review of SCHEMA_AUTHORITY.md before execution
- **Railway Deployment:** Git-based deployment with watchPatterns for controlled rollout
- **Database Scripts:** Never executed via railway run locally; uses public proxy URL

1.5 Testing Requirements

Code changes are tested before production deployment:

- **Type Checking:** TypeScript compilation required to pass
- **Linting:** ESLint validation enforced
- **Build Verification:** Turborepo build must complete successfully
- **Local Testing:** Functional testing in development environment before merge
- **Dependency Audit:** npm audit checked for known vulnerabilities

2. Cryptography

2.1 Data in Transit

All data transmitted between clients and servers is encrypted using TLS 1.2 or higher. This is enforced at the infrastructure level by all service providers:

Service	TLS Version	Enforcement
Vercel (Web Apps)	TLS 1.2+ (TLS 1.3 preferred)	Automatic; HTTPS enforced
Convex (Backend)	TLS 1.2+	Automatic; WebSocket over TLS
Railway (API/DB)	TLS 1.2+	Automatic; SSL required for Postgres
Clerk (Auth)	TLS 1.2+	Automatic; HTTPS only
Plaid API	TLS 1.2+	Required by Plaid

2.2 Data at Rest

All consumer data, including data received from the Plaid API, is encrypted at rest using industry-standard encryption:

Data Store	Encryption Standard	Key Management
Convex Database	AES-256 at rest	Provider-managed keys
Railway PostgreSQL	AES-256 at rest	Provider-managed keys
Backups	AES-256 encrypted	Provider-managed keys
Development Workstation	FileVault (XTS-AES-128)	Local keychain + recovery key

Plaid Data Handling: Consumer financial data received from the Plaid API is stored exclusively in encrypted databases (Convex or PostgreSQL). Access tokens are stored encrypted and are never logged or exposed in application code. Plaid webhooks are validated using signature verification before processing.

3. Logging and Monitoring

3.1 Audit Logging

The platform maintains comprehensive audit trails for material events across all production assets:

System	Events Logged	Retention
Clerk (Auth)	Login attempts, MFA events, session management, user lifecycle	Per Clerk retention policy
Convex	Function executions, database operations, errors, deployment events	7-30 days (dashboard)
Railway	API requests, database queries, deployment logs, errors	7 days (extendable)
Vercel	HTTP requests, function invocations, build logs, edge logs	1-30 days (by plan)
GitHub	Code changes, PR activity, actions runs, access events	90 days audit log
Neucleos Alerts	All security, availability, business, and operational alerts with acknowledgment tracking	Indefinite (Convex database)

3.2 Centralized Alerting System

A custom alerting system has been implemented within the Convex backend to provide centralized alert management, priority-based routing, and multi-channel notifications.

Alert Priority Levels:

Priority	Severity	Response Time	Notification Channels
P1	Critical	Immediate	SMS + Email + Slack
P2	High	< 1 hour	Email + Slack
P3	Medium	< 24 hours	Slack
P4	Low / Informational	Review daily	Dashboard only

3.3 Notification Infrastructure

The alerting system integrates with the following notification services:

Channel	Provider	Purpose
Slack	Slack Incoming Webhooks	Real-time team notifications for P1-P3 alerts with rich formatting
Email	Resend	Detailed alert notifications for P1-P2 with full context
SMS	Twilio	Immediate mobile notification for P1 critical alerts only
Dashboard	Convex + React	Centralized view of all alerts with acknowledgment tracking

3.4 Uptime Monitoring

External uptime monitoring is provided by BetterUptime, which monitors critical service endpoints and triggers alerts upon service degradation or outage:

- **Web Application:** Health endpoint monitored every 3 minutes
- **API Gateway:** Health endpoint monitored every 3 minutes
- **Incident Webhooks:** Automatic P1 alert creation when service goes down
- **Recovery Notifications:** P4 informational alert when service recovers

3.5 Alert Categories and Sources

Alerts are categorized and sourced as follows:

Category	Example Events	Source	Status
Security	Suspicious login, user deleted, failed auth	Clerk webhooks	**Active**
Availability	Service down, high latency, DB failure	BetterUptime, Convex	**Active**
Business	Plaid API errors, webhook failures	Plaid integration	**Pending***
Operational	Deploy failed, dependency vulnerabilities	GitHub, Vercel, Railway	**Active**

***Plaid Business Alerts:** Plaid API error alerting infrastructure is implemented and ready. Alert triggers will be activated upon receipt of full Plaid API production access.*

3.6 Alert Dashboard and Acknowledgment

All alerts are stored in the Convex database and accessible via a dedicated alerts dashboard that provides:

- Real-time view of active (unacknowledged) alerts
- Historical view of all alerts with filtering by priority and category
- Individual and bulk acknowledgment capabilities
- Notification delivery status tracking (Slack, Email, SMS sent indicators)

- Alert detail inspection with full context and metadata

4. Secrets Management

API keys, tokens, and credentials are managed securely:

- **Storage:** Secrets stored in provider secret management (Convex Dashboard, Railway Variables, Vercel Environment Variables)
- **Access:** Secrets accessed only server-side; never exposed to client code
- **Rotation:** API keys rotated periodically and immediately upon suspected compromise
- **Code Exclusion:** Secrets never committed to source control; .env files in .gitignore
- **Naming Convention:** Public-safe variables use VITE_* or NEXT_PUBLIC_* prefixes only

5. Review and Maintenance

This addendum is reviewed and updated:

- Upon implementation of new monitoring or alerting capabilities
- Quarterly: Verification of alert routing and notification delivery
- Annually: Full policy review aligned with main Information Security Policy