Certivo

SOC2 Compliance Documentation

Version: 1.1

Classification: Internal -- Auditor Reference

Last Updated: March 2026

CONFIDENTIAL *-- Not for distribution without written consent.*

Version: 2.0 **Last Updated:** 2026-03-17 (Added SMS, API, marketplace, forum, recurring invoice compliance sections) **Classification:** Internal — Auditor Reference **Master Roadmap:** `.claude/plans/master-roadmap.md` for prioritized security improvements

1. Trust Service Criteria Coverage

TSC	Category	Status	Implementation
CC6.1	Logical Access	Implemented	Firebase Auth + Custom Claims + Firestore Rules
CC6.2	Authentication	Implemented	Email/Password (all apps) + Ghost Mode
CC6.3	Authorization	Implemented	RBAC (5 roles) + TenantGuard + Firestore Rules
CC6.6	System Boundaries	Implemented	Multi-tenant org isolation via orgId
CC7.1	Change Management	Implemented	GitHub + CI/CD + PR checks
CC7.2	System Monitoring	Implemented	Cloud Logging + Error Reporter + Crash Logger
CC8.1	Incident Management	Implemented	Incident module + audit trail
A1.2	Availability	Partial	Firebase hosting (99.95% SLA), no custom SLA
PI1.1	Data Processing	Implemented	Server-side financial calculations

2. Authentication & Access Control

2.1 Authentication Methods

App	Method	MFA	Session
Tenant Suite	Email + Password (Firebase Auth)	TOTP (optional, admin-enforceable)	Browser session
Instructor Suite	Email + Password (Firebase Auth)	TOTP (optional)	Browser session + offline token
Client Portal	Email + Password (Firebase Auth)	Email OTP + trusted devices	sessionStorage
Super Admin	Email + Password + Ghost Mode	TOTP (optional)	Browser session

2.2 Custom Claims (Server-Enforced)

Custom claims are set by Cloud Functions and enforced by Firestore Security Rules:

```
initUserClaims → sets { orgId, role } on Firebase Auth token
initClientClaimsV2 → sets { orgId, role: 'client', clientId } on client auth token
ghostLogin → sets { orgId, role: 'ghost' } for tenant impersonation
```

Claims are synced on every login. Token refresh is forced after claim changes.

2.3 Role-Based Access Control

Role	Level	Permissions
SUPER_ADMIN	100	All operations, cross-tenant access
MANAGER	80	Financial access, user management, delete operations
SCHEDULER	50	Class scheduling, instructor assignment
STAFF	10	Read access, basic operations
client	—	Own data only, portal features

Roles are stored in Firestore user documents and synced to Auth custom claims. Firestore Security Rules enforce role checks on every read/write operation.

2.4 Rate Limiting

Function Category	Limit	Window	Implementation
Authentication (login)	5 attempts	15 minutes	Firestore-backed sliding window
Auth Claims	5 calls	5 minutes	enforceRateLimit()
Financial Operations	10 calls	1 minute	enforceRateLimit()
Data Modifications	20 calls	1 minute	enforceRateLimit()
Integrations	15 calls	1 minute	enforceRateLimit()
Utility	30 calls	1 minute	enforceRateLimit()

Rate limit state is persisted in Firestore `_rate_limits` collection. Stale entries are cleaned daily by the `cleanupRateLimits` scheduled function.

---

3. Data Isolation (Multi-Tenancy)

3.1 Tenant Scoping Architecture

Every document in tenant-scoped collections contains an `orgId` field. Three layers of enforcement:

1. **Client-side (TenantGuard):** Automatically adds `orgId` filter to all queries and stamps `orgId` on all writes via `TenantGuard.prepare()`.
2. **Server-side (Firestore Rules):** `isOrgMember()` helper validates that `request.auth.token.orgId == resource.data.orgId` on every read. `writesCorrectOrg()` validates orgId on every write.
3. **Cloud Functions (Admin SDK):** Financial calculations run server-side, scoped to the requesting user's org.

3.2 Cross-Tenant Access

Only Super Admins (role: `SUPER_ADMIN` or `super_admin`) and Ghost sessions bypass orgId checks. Ghost Mode creates a custom token with the target tenant's orgId, enabling controlled impersonation with full audit trail.

3.3 Collections with Tenant Scoping

All 28+ Firestore collections enforce orgId scoping except:

- `_rate_limits` — internal, no client access
- `system_messages` — global broadcast, read-only for clients
- `system_stats` — platform-wide, Super Admin read-only
- `admin_sessions` — Super Admin only
- `api_keys` — Super Admin only
- `automation_rules` — Super Admin only

---

4. Audit Trail

4.1 Audit Log Structure

All auditable events write to the `audit_logs` Firestore collection:

```
{
  "action": "RUN_PAYROLL",
  "userId": "uid_abc123",
  "orgId": "ORG_HEADQUARTERS",
  "details": { "month": 3, "year": 2026, "totalPay": 5420.00 },
  "timestamp": "2026-03-05T14:30:00.000Z",
  "ip": "request.rawRequest.ip"
}
```

4.2 Audited Operations

Action	Trigger	Details Captured
RUN_PAYROLL	runPayroll()	month, year, totalPay, classCount
GENERATE_CERTIFICATE	generateCertificate()	studentName, courseName, instructorName
GENERATE_DOCUMENT_{TYPE}	generateDocument()	type, storagePath
TRIGGER_DRIP_ENGINE	triggerDripEngine()	manual flag, triggeredBy, stats
ENROLL_IN_DRIP	enrollInDrip()	email, campaignName, enrolledBy
CREATE_USER	createUser()	email, role, orgId
GHOST_LOGIN	ghostLogin()	targetOrgId, impersonatorUid
CLIENT_LOGIN	initClientClaimsV2()	email, clientId
PROCESS_PAYOUT	processPayout()	instructorId, amount, status

4.3 Audit Log Protection

- Firestore Rules: `allow update: if false; allow delete: if false;`
- Audit logs are immutable once written
- Only admins and Super Admins can read audit logs
- Cloud Functions write via Admin SDK (bypasses security rules)

---

5. Logging & Monitoring

5.1 Client-Side Logging

Layer	Storage	Retention	Flush
Crash Logger (Black Box)	localStorage (`r4e_crash_log`)	100 entries	activateCloudLogging() → Cloud Function
Error Reporter	localStorage (`r4e_error_log`)	50 entries	Manual review / Sentry (when configured)

5.2 Server-Side Logging

Source	Destination	Retention
Cloud Functions	Google Cloud Logging	30 days (default)
Firestore Rules	Firebase console	30 days
Crash Logger flush	`logClientEvent` Cloud Function	Firestore (permanent)

5.3 Error Monitoring

The `error-reporter.js` module captures:

- Uncaught exceptions (`window.error` event)
- Unhandled promise rejections (`unhandledrejection` event)
- User context (userId, orgId) after authentication

Sentry integration is prepared but optional — set `VITE_SENTRY_DSN` to activate.

---

6. Data Protection

6.1 Data at Rest

- Firestore: Encrypted at rest by Google (AES-256)
- Cloud Storage: Encrypted at rest by Google (AES-256)
- No local database — all persistent data in Firebase

6.2 Data in Transit

- All Firebase connections use TLS 1.2+
- Cloud Functions enforce HTTPS
- No sensitive data in URL parameters

6.3 Secrets Management

Secret	Storage	Access
STRIPE_SECRET_KEY	Google Secret Manager	Cloud Functions only
STRIPE_WEBHOOK_SECRET	Google Secret Manager	Cloud Functions only
RESEND_API_KEY	Google Secret Manager	Cloud Functions only
Firebase API Key	Environment variable (.env)	Client-side (public, scoped by rules)

6.4 Storage Security Rules

- File uploads limited to 10MB
- Allowed types: images, PDF, CSV only
- Org-scoped paths (`/certificates/{orgId}/`, `/documents/{orgId}/`)
- Certificates and documents: write-only by Cloud Functions

---

7. Change Management

7.1 CI/CD Pipeline

```
Push to master → GitHub Actions → npm test → npm run build → firebase deploy
Pull Request → GitHub Actions → npm test → npm run build (no deploy)
```

7.2 Test Coverage

Suite	Framework	Count	Scope
Shared Library	Vitest + jsdom	250+ tests	auth, tenant-guard, utils, constants, cloud-functions, offline-queue, crash-logger, hooks, components
Cloud Functions	Vitest + Node	57 tests	Auth validation, rate limiting, payroll, PDF templates, tax calculations
E2E	Playwright	~47 tests	Login flows, UI rendering, accessibility (WCAG AA)
Total		**380 tests**	All critical paths covered

7.3 Deployment

- Firebase Hosting: 4 separate targets (one per app)
- Cloud Functions: northamerica-northeast2 (Montreal)
- Scheduled Functions: northamerica-northeast1 (for Cloud Scheduler compatibility)
- Firestore Rules: Deployed with every `firebase deploy`
- Storage Rules: Deployed with every `firebase deploy`

---

8. Incident Response

8.1 Incident Module

Built-in incident tracking in Tenant Suite:

- Incident creation with severity levels
- Photo/evidence uploads to Cloud Storage
- Automatic notification to admins
- Audit trail entry on creation

8.2 Support Tickets

Built-in support ticket system:

- Instructor-initiated tickets from field
- Admin triage and response
- Status tracking (open, in-progress, resolved)

---

9. Environment Separation

9.1 Configuration

Environment variables stored in `.env` files (git-ignored):

```
VITE_APP_ENV=development|staging|production
VITE_FIREBASE_PROJECT_ID=r4e-inventory
VITE_SENTRY_DSN=(optional)
```

9.2 Firebase Project Setup

Environment	Firebase Project	Status	Purpose
Development	r4e-inventory	■ Active	Active development + production
Staging	r4e-enterprise-staging	■ Not created	Pre-production testing
Production	r4e-inventory	■ Active	Live data (shared with dev currently)

Note: Development and production currently share the same Firebase project (`r4e-inventory`). Creating a separate staging project is documented in `docs/STAGING-SETUP.md`. A separate production project (`r4e-inventory-prod`) is recommended before accepting external tenants.

Separate Firebase projects ensure complete data isolation between environments.

---

10. Compliance Gaps & Remediation Status

Prioritized by: Security > Stability > Functionality > Revenue > Convenience > Visual **Full details:** `.claude/plans/master-roadmap.md`

■ Security Gaps — ■ ALL RESOLVED

Gap	Resolution	Roadmap Ref	Date
No input validation on forms	validateForm schemas on ALL 19 form-bearing views	SEC-1	2026-03-07
Hard deletes in Super Admin	Soft-delete pattern (status: 'archived') across all entities	SEC-2	2026-03-07
No password reset flow	sendPasswordResetEmail wired in all 3 email-auth apps	SEC-3	2026-03-07
No Client password reset	Client Portal uses Firebase Auth sendPasswordResetEmail + Tenant Suite reset button	SEC-4	2026-03-07
No MFA	TOTP-based MFA with QR setup, backup codes, admin enforcement	SEC-5	2026-03-08
Ghost Mode no tenant audit	ghost_sessions collection with full audit trail	SEC-6	2026-03-08
Session timeout not wired	useIdleTimeout + SessionWarningModal in all 4 apps	SEC-7	2026-03-08
Cloud Function orgId not validated	All 43 callable functions validate orgId from auth token	SEC-8	2026-03-07

■ Stability Gaps — ■ ALL RESOLVED

Gap	Resolution	Roadmap Ref	Date
No loading states in modals	ModalWrapper loading prop + all async modals use it	STB-1	2026-03-07
ErrorBoundary not per-view	ErrorBoundary wraps all React.lazy() views in all 4 apps	STB-2	2026-03-07
No write retry logic	firestoreRetry utility with exponential backoff (3 retries)	STB-3	2026-03-08
Inconsistent CF error shapes	wrapHandler() utility with standardized { code, message, details }	STB-4	2026-03-08
Firestore indexes incomplete	43 composite indexes deployed covering 28 collections	STB-5	2026-03-12

Infrastructure Gaps (Remaining)

Gap	Risk	Remediation	Effort
Single Firebase project	Medium	Create r4e-enterprise-staging (see docs/STAGING-SETUP.md)	1 hour
No penetration testing	Medium	Schedule annual pen test with third party	External
No data retention policy	Low	Define lifecycle + automate Firestore TTL	1 day
Cloud Logging retention (30 days)	Low	Export to BigQuery for long-term retention	0.5 days
No backup verification	Low	Automate Firestore export + restore tests	0.5 days

Resolved infrastructure items:

- ■ Firebase secrets deployed (2026-03-08)
- ■ Formal incident response plan documented (`docs/security/` + `docs/legal/incident-response-plan.md`)
- ■ Data breach response plan documented (`docs/legal/data-breach-response-plan.md`)

Security Improvements Timeline

Improvement	Date	Impact
Code splitting (React.lazy)	2026-03-06	Reduced attack surface per page load
Vendor chunking (7 named)	2026-03-06	Isolated vendor code for better CSP
useIdleTimeout hook + wiring	2026-03-06	Session timeout in all 4 apps
Canadian tax compliance (CPP/EI)	2026-03-06	Server-side financial accuracy
Input validation (all forms)	2026-03-07	SEC-1 complete
Cloud Function orgId hardening	2026-03-07	SEC-8 complete
MFA / TOTP enrollment	2026-03-08	SEC-5 complete
Ghost Mode audit trail	2026-03-08	SEC-6 complete
Firestore indexes deployed	2026-03-12	STB-5 complete
Ecosystem equalization (27 items)	2026-03-14	14 new views, ~5,300 lines
Landing page + dead code cleanup	2026-03-15	~300 lines removed, passive scroll listeners
Drip campaign rate controls	2026-03-06	Prevents email abuse
380 test coverage (up from 203)	2026-03-06	Broader regression detection

---

11. SMS / Twilio Compliance (CASL/TCPA)

Maps to: SOC2 PI1.1 (Data Processing)

11.1 Consent Collection

- **Double opt-in required:** Recipients must confirm consent via a follow-up confirmation message before receiving SMS notifications.
- Consent records stored in Firestore with timestamp, channel (SMS), and explicit confirmation flag.
- Separate opt-in per message category (class reminders, cert expiry alerts, shift confirmations, marketing).

11.2 Unsubscribe Handling

- Every SMS includes an unsubscribe keyword (STOP).
- Unsubscribe processed immediately upon receipt; maximum 2 business days for full removal across all systems.
- Unsubscribe events logged to `audit_logs` with timestamp and originating phone number (hashed).

11.3 Message Content Standards

- Clear sender identification on every message (business name, not short code alone).
- No sender ID spoofing — all messages originate from verified Twilio numbers.
- Message templates reviewed for compliance before activation.

11.4 Rate Limiting

- Maximum SMS per recipient per day enforced server-side (configurable per tenant, default: 5/day).
- Burst protection: Cloud Function rate limiting on `sendSMS` callable (10 calls/min per user).

11.5 Audit Trail

- Every SMS send logged: recipient (hashed), template ID, timestamp, delivery status, consent reference.
- Consent change events (opt-in, opt-out) logged with full chain of custody.

---

12. REST API Security

Maps to: SOC2 CC6.1-6.3 (Logical Access Controls)

12.1 API Key Management

- API keys generated per tenant via Super Admin or Tenant Suite settings.
- Keys are hashed (SHA-256) at rest in Firestore; plaintext shown once at creation.
- Key rotation supported: new key issued, old key remains valid for a configurable grace period (default: 72 hours).
- Revoked keys take effect immediately; no grace period on revocation.

12.2 Rate Limiting by Tier

Tier	Rate Limit	Burst
Starter	10 requests/min	20
Professional	100 requests/min	200
Enterprise	1,000 requests/min	2,000

Rate limit state tracked per API key in Firestore `_api_rate_limits` collection.

12.3 OAuth2 Scope Validation

- All API requests scoped to the tenant's `orgId` — no cross-tenant data access.
- OAuth2 scopes define granular permissions (e.g., `classes:read`, `invoices:write`, `students:read`).
- Scope validation enforced at the Cloud Function layer before any Firestore operation.

12.4 Webhook Security

- Outbound webhooks signed with HMAC-SHA256 using a per-tenant webhook secret.
- Webhook secret rotatable independently of API keys.
- Delivery attempts logged with timestamp, endpoint URL, response status, and retry count.
- Failed deliveries retried with exponential backoff (max 5 retries over 24 hours).

---

13. Marketplace Payment Security

Maps to: SOC2 PI1.1 (Data Processing), CC6.1 (Logical Access)

13.1 PCI DSS Compliance

- **No card data stored on Certivo infrastructure.** All payment flows routed through Stripe (PCI DSS Level 1 certified).
- Stripe Checkout Sessions used for all marketplace purchases — card data never touches Certivo servers.
- Stripe Connect handles instructor payouts with platform fee deduction.

13.2 Payout Idempotency

- Every payout operation uses an idempotency token (generated from `instructorId + period + timestamp`).
- Duplicate payout attempts within the idempotency window are rejected with a clear error.
- Idempotency tokens stored in `payout_idempotency` Firestore collection with 30-day TTL.

13.3 Refund & Dispute Handling

- 14-day refund window for marketplace purchases.
- Disputes escalated to platform admin via `marketplace_disputes` collection.
- Dispute resolution audit trail: creation, evidence submission, resolution, and payout adjustment.

13.4 Payment Reconciliation

- Every marketplace transaction generates a reconciliation record linking Stripe payment ID, platform fee, instructor payout, and tenant commission.
- Monthly reconciliation reports available in Super Admin.
- Discrepancies flagged automatically when Stripe webhook totals diverge from Firestore records.

---

14. Community Forum Compliance

Maps to: SOC2 A1.2 (Availability), CC6.6 (System Boundaries)

14.1 Tenant Data Isolation

- All forum content scoped to `orgId` — posts, comments, and user profiles are tenant-isolated.
- Cross-tenant forum access is prohibited; Super Admin can view but not post across tenants.

14.2 Content Moderation

- Moderation actions (hide, delete, warn, ban) logged to `audit_logs` with moderator ID, action type, content reference, and reason.
- Moderation SLA: flagged content reviewed within 24 hours (configurable per tenant).
- Automated spam prevention via rate limiting on post creation (max 10 posts/hour per user).

14.3 User Reporting

- Users can report posts/comments; reports queued in `forum_reports` collection.
- Report response SLA tracked per tenant (default: 48 hours).
- Reporter identity protected from the reported user.

14.4 Spam Prevention

- Post creation rate limited: 10 posts/hour, 50 posts/day per user.
- Link posting restricted for accounts less than 7 days old.
- Duplicate content detection on consecutive posts.

---

15. Recurring Invoice Automation

Maps to: SOC2 PI1.1 (Data Processing), CC8.1 (Incident Management)

15.1 Idempotent Invoice Generation

- Each scheduled invoice generation uses an idempotency key derived from `clientId + schedule + billing period`.
- Duplicate invoice creation within the same billing period is prevented at the Cloud Function layer.
- Invoice generation events logged to `audit_logs` with schedule reference and generation timestamp.

15.2 Failed Payment Notification

- Failed payment attempts trigger automatic notification to both the tenant admin and the client.
- Retry schedule: 1 day, 3 days, 7 days after initial failure (configurable per tenant).
- After final retry failure, invoice status set to `payment_failed` and escalated to tenant admin dashboard.

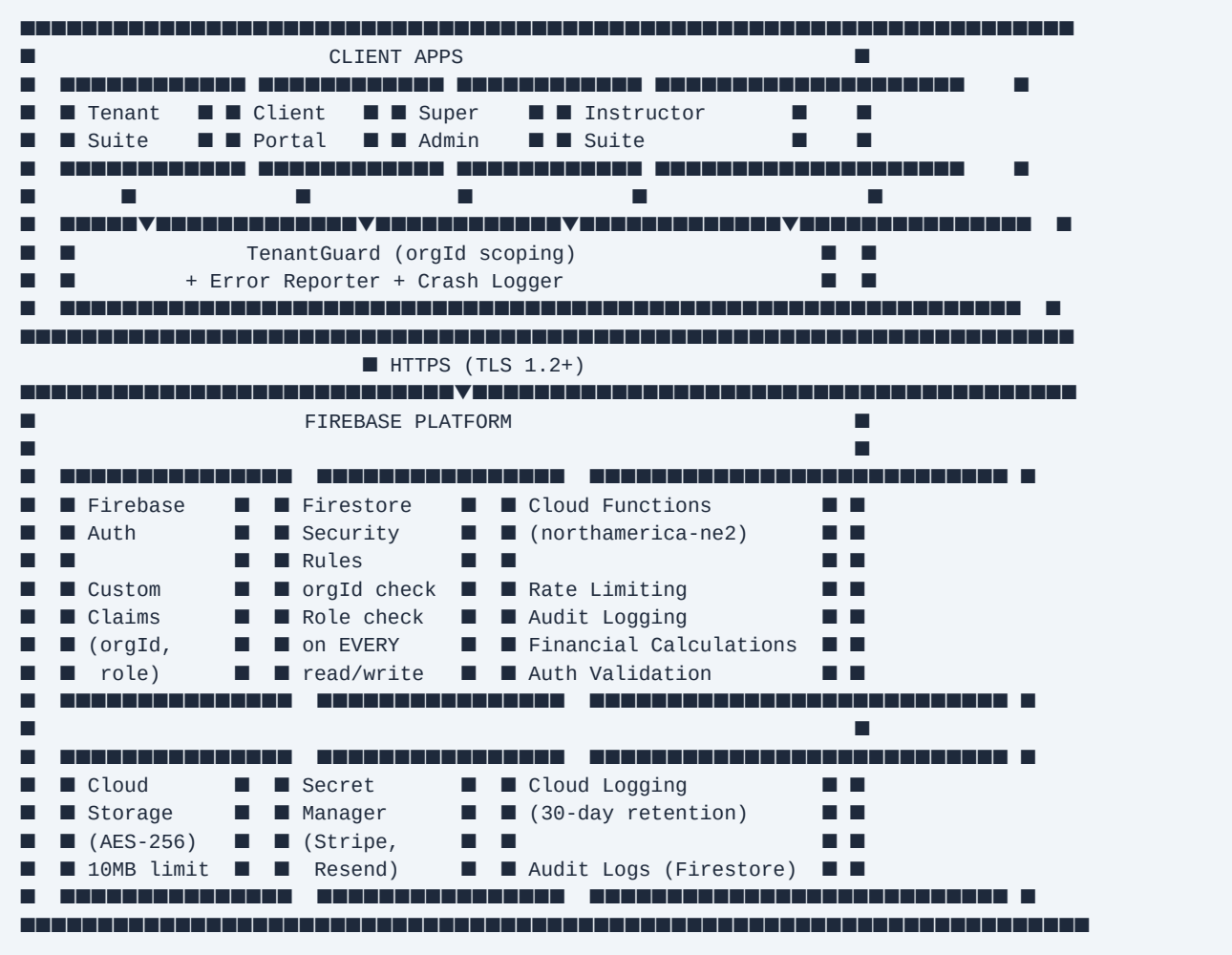
15.3 Customer Verification

- Recurring invoice schedules require explicit client acknowledgment before activation.
- Schedule changes (amount, frequency, line items) generate a change notification to the client.
- Clients can pause or cancel recurring invoices through the Client Portal.

15.4 Audit Trail

- Every recurring invoice event logged: schedule creation, invoice generation, payment attempt, payment success/failure, schedule modification, schedule cancellation.
- Immutable audit records — same protection as all `audit_logs` entries (no update, no delete).

Appendix A: Security Architecture Diagram



Appendix B: Firestore Collection Security Matrix

Collection	Read	Create	Update	Delete
users	self + org	admin/manager	self + admin	super_admin
clients	org + client	org write	org + client	admin
classes	org	org write	org	admin
invoices	org	org write	org	admin
payouts	org	org write	admin	NEVER
audit_logs	org + super	auth	NEVER	NEVER
tenants	super + self	super	super	super
_rate_limits	NEVER	NEVER	NEVER	NEVER

("org" = orgId match required, "super" = SUPER_ADMIN only, "NEVER" = immutable)