ComplianceSuite

FIT-CLEAN

CS-DOC-0003 · COMPLIANCESUITE DOCUMENTATION

# Roles and Permissions.

FIT-only redaction. Effective 2026-04-28.

| DOCUMENT ID | VERSION | EFFECTIVE | OWNER |
|---|---|---|---|
| CS-DOC-0003 | v1.0 | 2026-04-28 | Customer Success |

*Public — Documentation · Review cycle: On change*

01 — DOCUMENT CONTROL

# Control block and metadata anchor.

The control block identifies the document, its current revision, the regulated process it supports, and the people accountable for its lifecycle. Every value below is the source of truth for any downstream record, audit trail entry, or signature block.

| | |
|---|---|
| DOCUMENT ID | CS-DOC-0003 |
| TITLE | Roles and Permissions |
| VERSION | v1.0 |
| STATUS | FIT-CLEAN |
| EFFECTIVE DATE | 2026-04-28 |
| REVIEW CYCLE | On change |
| DOCUMENT OWNER | Customer Success |
| CLASSIFICATION | Public — Documentation |
| RELATED RECORDS | /output/CS-DOC-0003_Roles_and_Permissions.pdf |
| SUPERSEDES | — (initial release) |

02 — APPROVALS

# Sign-off table, ready for ink or e-signature.

The signatures below confirm review and authorisation of this document. Approvals must be recorded in chronological order. If the document is signed electronically, the e-signature record on the ComplianceSuite platform supersedes any handwritten entry on this page and carries the same legal weight under 21 CFR Part 11 and EU GMP Annex 11.

| Role | Name | Function | Date | Signature |
|---|---|---|---|---|
| Author | | Validation Lead | | |
| Reviewer | | Quality Assurance | | |
| Reviewer | | Process / System Owner | | |
| Approver | | Head of Quality | | |
| Approver | | Regulatory Affairs | | |

03 — CONTENTS

# What's in this document.

01 — WHAT THIS EDITION COVERS

# What this edition covers.

This documentation covers ComplianceSuite's permission model:

- The 3-level permission concept (account/tenant/change)
- Role structure with permissions and isCustom flag
- Hardwired separation of duties (SoD) enforcement
- TenantUserAssignment with roleId for tenant-scoped assignments
- isSystemAdmin and isAccountAdmin flags on user

# What this edition does NOT cover (Roadmap topics).

The following concepts from the original spec are not or only partly implemented:

- **Complete built-in roles taxonomy with concrete names** — role model exists, but a defined list of named built-in roles (account owner, account admin, tenant owner, QA approver, etc.) not enforced in code. (Roadmap: see `/gap-implementation-plans/PLAN-04-*.md`)
- **Custom roles must inherit from built-in + narrow only** — isCustom flag exists on role, but no validator enforces inheritance or narrowing-only rule.
- **Permission resolution algorithm "lowest-grant-wins"** — no implemented resolver logic cascading inheritance and overrides across all four levels.
- **Delegation with time limit** — not implemented.
- **Account-admin/tenant-admin/system-admin bypass logic** — flags exist, but bypass mechanisms not systematically documented.

03 — HOW PERMISSIONS WORK

# How permissions work.

ComplianceSuite enforces every authorization decision against a single, deterministic question:

> **NOTE**
> "At the lowest level where this user has been granted a value — what does that value say?"

The answer is the user's effective permission. There are no implicit assignments, no overlap rules, no priority order to remember — just the 4-level hierarchy (account → tenant → system → change) and the rule that the lowest explicit assignment wins.

## The three components of every assignment

| Component | What it specifies | Examples |
|---|---|---|
| Subject | The individual user granted access | `jane.doe@acme.com` (email as unique user ID) |
| Role | The named bundle of capabilities | Tenant owner, QA approver, system author, read-only auditor, custom role |
| Scope | The hierarchy node where the role applies | Account, a tenant, a system, a single change |

**Effective permission = lowest explicit assignment:**

If Jane is **QA approver** at tenant level and **read-only** on a specific system, her effective permission on that system is **read-only**. Platform always narrows on the way down, never expands. There is no configuration setting to reverse this — intentionally.

04 — PERMISSION PATTERNS IN THIS IMPLEMENTATION

# Permission patterns in this implementation.

## The 3-level model

ComplianceSuite implements permission guards at three levels:

| Level | Guard function | Purpose |
|---|---|---|
| Account level | `requireAccountPermission()` | Account admin, user management, tenant lifecycle |
| Tenant level | `requireTenantPermission()` | Tenant-specific operations, system registration, change management |
| Change level | `requireChangePermission()` | Change-specific operations, deliverable approval |

**System admin and account admin bypass:**

Users with `isSystemAdmin = true` or `isAccountAdmin = true` bypass all permission checks at their respective level.

## Role model

The `role` model in Prisma has:

```
model Role {
  id            String
  name          String
  isCustom      Boolean @default(false)  // true if customer-authored
  permissions   String[]                // array of capability strings
  accountId     String
  account       Account
  // ...
}
```

The `permissions` are modeled as string array. No resolver logic implemented for inheritance or narrowing validation.

## TenantUserAssignment

The M2M relationship between user and tenant carries an optional `roleId`:

```
model TenantUserAssignment {
  id        String
  userId    String
  tenantId  String
  roleId    String?    // optional tenant-scoped role override
  isDefault Boolean    // default tenant on login
  // ...
}
```

This allows assigning a user to a tenant with a specific role.

# Separation of Duties (SoD) — Hardwired Enforcement.

ComplianceSuite enforces SoD at change level. The enforcement point is in the signature and approval flows:

| SoD Rule | Status in code | Enforced in |
|---|---|---|
| Author ≠ Reviewer | Implemented | `app/actions/signatures.ts:35` |
| Author ≠ Approver | Implemented | `app/actions/signatures.ts:44` |
| Reviewer ≠ Approver | Configurable (default disabled) | Policy engine (planned) |

**Why SoD is hardwired:**

Every published warning letter concerning computerized systems contains at least one deviation of the form: *"the same person approved their own work"*. Hardwiring SoD removes the most common Annex 11 / Part 11 deviation category before inspection starts.

06 — USER FLAGS FOR ADMIN ACCESS

# User Flags for Admin Access.

The `user` model carries two admin flags:

```
model User {
  id               String
  email            String @unique
  isSystemAdmin    Boolean @default(false)  // platform-wide admin
  isAccountAdmin   Boolean @default(false)  // account-level admin (NEW)
  // ...
}
```

- **isSystemAdmin** — bypasses all permission checks. Access to every account, every tenant, every system.
- **isAccountAdmin** — bypasses permission checks at account level. Access to account settings, user management, tenant creation within account.

07 — PERMISSION TROUBLESHOOTING

# Permission Troubleshooting.

| Symptom | Likely Cause | Resolution |
|---|---|---|
| User cannot see tenant expected | No tenant-scoped role assigned; account-level role alone grants no tenant visibility | Assign tenant role (read-only is minimum) |
| User can read but not write | Effective permission is lower of two assignments | Check system-level assignments narrower than tenant-level |
| Approval button grayed out | User is author or reviewer of same record (SoD) | Reassign author/reviewer; platform will not relax SoD |
| Custom role not available | Role not yet QA-approved or was retired | Account compliance lead must approve, or role must be revived |

# Code Reference.

## Prisma Models

- `User` — with `isSystemAdmin`, `isAccountAdmin`, `tenantAssignments` M2M
- `Role` — with `name`, `isCustom`, `permissions` (string[]), `accountId`
- `TenantUserAssignment` — M2M with `userId`, `tenantId`, optional `roleId`
- `Account` — with `roles` relation for account-level roles

## Server Actions

- `app/actions/account/*` — account creation, user management
- `app/actions/tenant/*` — tenant configuration, user assignments
- `app/actions/change/*` — change lifecycle with SoD enforcement in approval gates
- `app/actions/signatures.ts` — e-signature workflows with SoD checks at lines 35, 44

## Permission Guards (planned / partly implemented)

- `requireAccountPermission()` — guard for account-level operations
- `requireTenantPermission()` — guard for tenant-level operations
- `requireChangePermission()` — guard for change-level operations

**End of documentation**

REVISION HISTORY

# Every change, tracked and signed.

Add one row for every controlled revision. Minor changes (typos, formatting) increment the patch version; substantive edits trigger a fresh review cycle and a new approver round.

| Version | Date | Author | Summary of Change | Approver |
|---|---|---|---|---|
| 1.0 | 2026-04-28 | Documentation Team | FIT-only redaction limited to codebase-verified functionality. | Head of Documentation |
| — | — | — | Reserved for next revision. Do not delete this row. | — |

GLOSSARY

# Shared language, no ambiguity.

Definitions used throughout this document. Where a term has a specific meaning inside ComplianceSuite, the platform-specific definition takes precedence over the generic regulatory term.

| | |
|---|---|
| **CSV** | Computerized Systems Validation |
| **GAMP 5** | Good Automated Manufacturing Practice, Edition 5 (2nd edition, 2022) |
| **GxP** | Good 'x' Practice — covers GMP, GLP, GCP, GDP, GVP |
| **IQ / OQ / PQ** | Installation / Operational / Performance Qualification |
| **Part 11** | 21 CFR Part 11 — US FDA rule on electronic records and electronic signatures |
| **Annex 11** | EU GMP Annex 11 — EU rule on computerised systems |
| **URS** | User Requirements Specification |
| **FRS** | Functional Requirements Specification |
| **RTM** | Requirements Traceability Matrix |
| **SOP** | Standard Operating Procedure |
| **ALCOA+** | Attributable, Legible, Contemporaneous, Original, Accurate (+ Complete, Consistent, Enduring, Available) |
| **ICH Q9** | International Council for Harmonisation Quality Risk Management guideline |

— End of document —