

CS-DOC-0010 · COMPLIANCESUITE DOCUMENTATION

Periodic Review.

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

DOCUMENT ID	VERSION	EFFECTIVE	OWNER
CS-DOC-0010	v1.0	2026-04-28	Customer Success

Public — Documentation · Review cycle: On change

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-0010
TITLE	Periodic Review
VERSION	v1.0
STATUS	FIT-CLEAN
EFFECTIVE DATE	2026-04-28
REVIEW CYCLE	On change
DOCUMENT OWNER	Customer Success
CLASSIFICATION	Public — Documentation
RELATED RECORDS	Generated from /content/doc_review.py
SUPERSEDES	— (initial release)

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		

What's in this document.

01 — Document Control	—
02 — Approvals	—
03 — Contents	—
01 — What this edition covers	—
02 — What this edition does NOT cover	—
03 — PeriodicReview — Model and Data Structure	—
04 — ReviewNumber Generator	—
05 — ReviewChecklist — Generic Sub-Model	—
06 — Server Actions	—
07 — Conclusion Field	—
08 — Pages	—
09 — Findings and Corrective Actions — Generic Json Fields	—
10 — Code Reference	—
Revision History	—

Glossary & Abbreviations

.....
.....

—

What this edition covers.

This documentation describes the Periodic Review Model in the ComplianceSuite codebase:

- **PeriodicReview:** Data model with Fields: reviewNumber, reviewPeriod, status, scheduledDate, startedAt, completedAt, findings, correctiveActions, conclusion, nextReviewDate
- **reviewPeriod Values:** ANNUAL, SEMI_ANNUAL, QUARTERLY
- **ReviewChecklist:** Generic Sub-Model for Checklist Items
- **Server Actions:** Implemented in `app/actions/system/periodic-review.ts`
- **Pages:** `app/periodic-review/page.tsx` and `app/periodic-review/[id]/page.tsx`
- **Review Number Generator:** Generates PR-YYYY-NNN pattern (Line 10–27)
- **Cadence Logic:** Calculates nextReviewDate based on reviewPeriod

What this edition does **NOT** cover.

This edition does NOT describe the following functions intended in the original spec, because they are not implemented in the code:

- **Auto-Population of Review Report** – No automation for Report creation
- **5-Section Mandatory Structure** – Use/Scope, Config Baseline, Risk Re-Eval, Deviations/CAPAs, Access Review are not enforced; `findings` is generic Json
- **Auto Audit Trail Digest** – No automated Digest for Review Window
- **Outcome Enum** – `REMAINS_VALIDATED/DRIFT_DOCUMENTED/CONFIGURATION_CHANGE_SCHEDULED` etc. are not coded; `conclusion` is free-form String
- **Cadence Protection (tighten only)** – No Validation for Cadence Changes
- **Auto Trigger Configuration/Upgrade/Decommissioning Changes** – No automatic Change Creation
- **Risk Reassessment Obligation** – No Validator for Risk Re-assessment

PeriodicReview — Model and Data Structure.

Table Definition

The table `PeriodicReview` (Prisma Schema, Line 1094–1122) stores:

Field	Type	Description
<code>id</code>	String UUID	Review ID
<code>systemId</code>	String UUID (FK)	System assignment
<code>reviewNumber</code>	String @unique	PR-YYYY-NNN (e.g., "PR-2026-001")
<code>reviewPeriod</code>	String	ANNUAL | SEMI_ANNUAL | QUARTERLY (Default: ANNUAL)
<code>status</code>	String	SCHEDULED | IN_PROGRESS | COMPLETED | OVERDUE
<code>scheduledDate</code>	DateTime	Planned Review start date
<code>startedAt</code>	DateTime?	Actual start time
<code>completedAt</code>	DateTime?	Completion time
<code>reviewerId</code>	String UUID (FK)	Reviewer (User)
<code>approvedById</code>	String UUID? (FK)	Approver (User, optional)
<code>findings</code>	Json?	Findings/Checklist Results (generic Json)
<code>correctiveActions</code>	Json?	Corrective Actions (generic Json)
<code>conclusion</code>	String?	Conclusion (free-form String, not typed)
<code>nextReviewDate</code>	DateTime?	Next Review due on
<code>tenantId</code>	String UUID?	Tenant assignment (Shadow Field)
<code>organizationId</code>	String UUID (FK)	Organization assignment (Legacy)

ReviewPeriod Values

```
ANNUAL      → 12 months
SEMI_ANNUAL → 6 months
QUARTERLY   → 3 months
```

Code Reference (Line 31–46, `periodic-review.ts`):

```
function calculateNextReviewDate(completedAt: Date, reviewPeriod: string): Date {
  const next = new Date(completedAt);
  switch (reviewPeriod) {
    case 'QUARTERLY':
      next.setMonth(next.getMonth() + 3);
      break;
    case 'SEMI_ANNUAL':
      next.setMonth(next.getMonth() + 6);
      break;
    case 'ANNUAL':
    default:
      next.setFullYear(next.getFullYear() + 1);
      break;
  }
  return next;
}
```

ReviewNumber Generator.

The function `generateReviewNumber()` (Line 10–27, `periodic-review.ts`) generates unique Review Numbers following pattern **PR-YYYY-NNN**:

```

async function generateReviewNumber(): Promise {
  const db = await getTenantScopedDb();
  const year = new Date().getFullYear();
  const prefix = `PR-${year}-`;

  const lastReview = await db.periodicReview.findFirst({
    where: { reviewNumber: { startsWith: prefix } },
    orderBy: { reviewNumber: 'desc' },
  });

  let nextNum = 1;
  if (lastReview) {
    const lastNum = parseInt(lastReview.reviewNumber.split('-')[2], 10);
    nextNum = lastNum + 1;
  }

  return `${prefix}${String(nextNum).padStart(3, '0')}`;
}

```

Examples:

- 2026 Review 1: PR-2026-001
- 2026 Review 42: PR-2026-042
- 2027 Review 1: PR-2027-001 (Year changes, counter resets)

The `reviewNumber` is Unique Constraint, so each number can occur only once.

ReviewChecklist — Generic Sub-Model.

The table `ReviewChecklist` (Prisma Schema, Line 1124–1134) stores Checklist Items for a Review:

Field	Type	Description
<code>id</code>	String UUID	Item ID
<code>reviewId</code>	String UUID (FK)	Review assignment
<code>category</code>	String	Category (e.g., CHANGE_CONTROL, INCIDENTS, ACCESS, etc.)
<code>item</code>	String	Checklist Item (free-form Text)
<code>result</code>	String	SATISFACTORY | NEEDS_IMPROVEMENT | NON_COMPLIANT | NOT_APPLICABLE
<code>evidence</code>	String?	Optional: Evidence
<code>notes</code>	String?	Optional: Notes

Semantics: The `category` is a String (not an Enum) and can be arbitrary. The `result` follows a predefined list.

Note: The Checklist structure is generic; there is no enforced 5-Section structure in code.

Server Actions.

scheduleReview(params: { systemId, reviewPeriod?, scheduledDate })

Purpose: Schedule a new Periodic Review.

Parameters:

- `systemId` – System against which the Review runs
- `reviewPeriod` – Optional (ANNUAL, SEMI_ANNUAL, QUARTERLY); Default: ANNUAL
- `scheduledDate` – Planned start date (ISO String)

Flow:

- 01 Generate `reviewNumber` with `generateReviewNumber()`
- 02 Create `PeriodicReview` with Status SCHEDULED
- 03 Write `AuditLog` entry
- 04 Revalidate `/periodic-review` pages

Code Reference (Line 50–94, `periodic-review.ts`):

```

export async function scheduleReview(params: {
  systemId: string;
  reviewPeriod?: string;
  scheduledDate: string;
}) {
  const user = await getCurrentUser();
  if (!user) throw new Error('Unauthorized');

  const tenantId = await getActiveTenantId();
  const db = await getTenantScopedDb();
  const reviewNumber = await generateReviewNumber();

  const review = await db.periodicReview.create({
    data: {
      systemId: params.systemId,
      reviewNumber,
      reviewPeriod: params.reviewPeriod || 'ANNUAL',
      scheduledDate: new Date(params.scheduledDate),
      reviewerId: user.id,
      tenantId,
      status: 'SCHEDULED',
    },
  });

  // ... Create Audit Log ...

  revalidatePath('/periodic-review');
  return review;
}

```

startReview(reviewId)

Starts a planned Review (sets `startedAt` and changes Status to `IN_PROGRESS`).

(Line 98ff., `periodic-review.ts` – not fully read, but present)

completeReview(reviewId, data)

Completes a Review (sets `completedAt`, `conclusion`, `findings`, `correctiveActions`, calculates `nextReviewDate`).

Additional Server Actions

Additional Review Management functions are in the file but were not fully read.

Conclusion Field.

The Field `conclusion` (String?, Prisma Schema Line 1110) is free-form Text and not typed. In the original, 5 specific Outcomes are provided:

- REMAINS_VALIDATED
- DRIFT_DOCUMENTED
- CONFIGURATION_CHANGE_SCHEDULED
- UPGRADE_CHANGE_SCHEDULED
- DECOMMISSIONING_CHANGE_SCHEDULED

In code there is no enforced Enum list for these Outcomes. The `conclusion` can be arbitrary.

Pages.

/periodic-review/page.tsx

Overview page of all Periodic Reviews (Tenant level).

/periodic-review/[id]/page.tsx

Detail page for a specific Periodic Review with Edit functions.

Findings and Corrective Actions — Generic **Json Fields.**

The Fields `findings` and `correctiveActions` are both type `Json`? (Prisma Schema Lines 1108–1109). There is no Schema enforcement for their structure:

- `findings` – Can contain arbitrary `Json` (typically Checklist Results)
- `correctiveActions` – Can contain arbitrary `Json` (typically identified Actions)

Code does not presume structure; the Application is responsible for consistent `Json` structure.

Code Reference.

Primary Files:

- `app/actions/system/periodic-review.ts` – Server Actions for Review Management
- `prisma/schema.prisma` (Line 1094–1122) – PeriodicReview Model
- `prisma/schema.prisma` (Line 1124–1134) – ReviewChecklist Model
- `app/periodic-review/page.tsx` – Overview page
- `app/periodic-review/[id]/page.tsx` – Detail page
- `app/actions/periodic-review.ts` – Additional Legacy Actions (compat)

Test Coverage:

- `app/actions/system/__tests__/periodic-review.test.ts`
- `tests/e2e/interactions/periodic-review.spec.ts`

Line count: 279 lines **Date:** 2026-04-28

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 —