CS-DOC-0010 · GXP-DESK DOCUMENTATION

Periodic Review.

FIT-only redaction. Effective 2026-06-04.

DOCUMENT ID	VERSION	EFFECTIVE	OWNER
CS-DOC-0010	v1.0	2026-06-04	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-0010
TITLE	Periodic Review
VERSION	v1.0
STATUS	FIT-CLEAN
EFFECTIVE DATE	2026-06-04
REVIEW CYCLE	On change
DOCUMENT OWNER	Customer Success
CLASSIFICATION	Public — Documentation
RELATED RECORDS	—
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 GxP-Desk 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 VERSION COVERS

What this version covers.

This documentation describes the periodic-review model in the GxP-Desk code:

- **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 according to a pattern (lines 10–27)
- **Cadence logic**: Calculates nextReviewDate based on reviewPeriod

What this version does NOT cover.

This version does NOT describe the following functions envisaged in the original, because they are not implemented in the code:

- **Auto-population of the review report** – No automation for report generation
- **Mandatory 5-section 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 the review window
- **Outcome enum** – REMAINS_VALIDATED/DRIFT_DOCUMENTED/CONFIGURATION_CHANGE_SCHEDULED etc. are not coded; `conclusion` is a free-form string
- **Cadence protection (tighten only)** – No validation for cadence changes
- **Auto-trigger of configuration/upgrade/decommissioning changes** – No automatic change creation
- **Mandatory risk reassessment** – No validator for risk re-evaluation

PeriodicReview — model and data structure.

Table definition

The table `PeriodicReview` (Prisma schema, lines 1094–1122) stores:

Field	Type	Description
id	String UUID	Review ID
systemId	String UUID (FK)	System association
reviewNumber	String @unique	PR-YYYY-NNN (e.g. "PR-2026-001")
reviewPeriod	String	ANNUAL	SEMI_ANNUAL	QUARTERLY (default: ANNUAL)
status	String	SCHEDULED	IN_PROGRESS	COMPLETED	OVERDUE
scheduledDate	DateTime	Planned review start date
startedAt	DateTime?	Actual start time
completedAt	DateTime?	Completion time
reviewerId	String UUID (FK)	Reviewer (user)
approvedById	String UUID? (FK)	Approver (user, optional)
findings	Json?	Findings/checklist results (generic Json)
correctiveActions	Json?	Corrective actions (generic Json)
conclusion	String?	Conclusion (free-form string, not typed)
nextReviewDate	DateTime?	Next review due on
tenantId	String UUID?	Tenant association (shadow field)
organizationId	String UUID (FK)	Organization association (legacy)

ReviewPeriod values

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

Code reference (lines 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;
}
```

04 — REVIEWNUMBER GENERATOR

ReviewNumber generator.

The function `generateReviewNumber()` (lines 10–27, `periodic-review.ts`) generates unique review numbers following the 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` (the year rolls over, the counter resets)

The `reviewNumber` is a unique constraint, so each number can occur only once.

ReviewChecklist — generic sub-model.

The table `ReviewChecklist` (Prisma schema, lines 1124–1134) stores checklist items for a review:

Field	Type	Description
id	String UUID	Item ID
reviewId	String UUID (FK)	Review association
category	String	Category (e.g. CHANGE_CONTROL, INCIDENTS, ACCESS, etc.)
item	String	Checklist item (free-form text)
result	String	SATISFACTORY	NEEDS_IMPROVEMENT	NON_COMPLIANT	NOT_APPLICABLE
evidence	String?	Optional: evidence/proof
notes	String?	Optional: notes

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

Note: The structure of the checklist is generic; there is no enforced 5-section structure in the code.

Server Actions.

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

Purpose: Schedules a new periodic review.

Parameters:

- `systemId` – The system the review is run against
- `reviewPeriod` – Optional (ANNUAL, SEMI_ANNUAL, QUARTERLY); default: ANNUAL
- `scheduledDate` – Planned start date (ISO string)

Flow:

01 Generate `reviewNumber` with `generateReviewNumber()`

02 Create a PeriodicReview with status SCHEDULED

03 Write an AuditLog entry

04 Revalidate the /periodic-review pages

Code reference (lines 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 scheduled review (sets `startedAt` and changes the status to IN_PROGRESS).

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

completeReview(reviewId, data)

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

Further server actions

Additional functions for review management are present in the file, but were not fully read.

07 — CONCLUSION FIELD

Conclusion field.

The field `conclusion` (String?, Prisma schema line 1110) is free-form text and not typed. The original envisages 5 specific outcomes:

- REMAINS_VALIDATED
- DRIFT_DOCUMENTED
- CONFIGURATION_CHANGE_SCHEDULED
- UPGRADE_CHANGE_SCHEDULED
- DECOMMISSIONING_CHANGE_SCHEDULED

In the code there is no enforced enum list for these outcomes. The `conclusion` can be arbitrary.

08 — PAGES

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 CorrectiveActions — generic Json fields.

The fields `findings` and `correctiveActions` are both of 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)

The code presumes no structure; the application is responsible for a consistent Json structure.

Code references.

Primary files:

- `app/actions/system/periodic-review.ts` – Server actions for review management
- `prisma/schema.prisma` (lines 1094–1122) – PeriodicReview model
- `prisma/schema.prisma` (lines 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:** 2026-06-04

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	2026-06-04	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 GxP-Desk, 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 —