CS-DOC-0009 · GXP-DESK DOCUMENTATION

Approvers and Separation of Duties.

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

DOCUMENT ID	VERSION	EFFECTIVE	OWNER
CS-DOC-0009	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-0009
TITLE	Approvers and Separation of Duties
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 signature and approval model in the GxP-Desk code:

- **ElectronicSignature**: 3-signature model (Author → Reviewer → Approver) with meaning strings
- **Signature Meanings**: AUTHORED, REVIEWED, APPROVED, REJECTED (hard-coded)
- **SignatureWorkflow**: Sequential or parallel (workflowType)
- **SignatureStep**: Step-by-step signature tracking with status and date
- **Separation of Duties (SoD)**: Enforcement via `separationOfDutiesEnabled` (Boolean, default: true) at the customer/tenant level
- **Server Actions**: Implemented in `app/actions/signatures.ts`
- **Re-authentication on signature** (NEW 2026-06-04): Step-up via MFAGrant (single-use, 5-minute window) is enforced on submit/approve/reject; `passwordVerified` reflects the completed re-authentication

02 — WHAT THIS VERSION DOES NOT COVER

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:

- **Deterministic routing** – The "lowest scoped role wins" algorithm is not implemented
- **Parallel review/approval for multi-sig** – `workflowType` exists (SEQUENTIAL/PARALLEL), but the UI/enforcement is immature
- **SoD evidence export** – Per-record SoD export with role snapshots is not coded
- **Time-limited delegation** – Controlled delegation with a delegation window and Account Compliance Lead control is not implemented
- **Signature Meaning Library** – Custom meanings are not provided for; only the 4 hard-coded values

ElectronicSignature — 3-signature model.

Data structure

The table `ElectronicSignature` (Prisma schema, lines 596–623) stores each signature:

Field	Type	Description
id	String UUID	Signature ID
documentId	String UUID (FK)	Association with a document
documentVersion	String	Version of the signed document
signerId	String UUID (FK)	Signer (user)
signerName	String	Name of the signer (snapshot)
signerEmail	String	Email of the signer (snapshot)
signerTitle	String?	Title/role (optional)
meaning	String	AUTHORED	REVIEWED	APPROVED	REJECTED
signedAt	DateTime	Timestamp of the signature
timezone	String	Time zone of the signer (default: UTC)
passwordVerified	Boolean	Default `true`
contentHash	String	SHA-256 of the content at signing time
signatureHash	String	SHA-256 of the signature itself
ipAddress	String?	IP address of the signer
userAgent	String?	Browser/client info
isValid	Boolean	Is this signature valid? (Invalidation possible)
invalidatedAt	DateTime?	When did the signature become invalid?
invalidationReason	String?	Reason for invalidation

Field	Type	Description
comments	String?	Optional: comments from the signer
workflowStepId	String UUID? (Unique)	Association with a SignatureStep

Signature Meanings

The four meanings are hard-coded (no custom meanings):

Meaning	Semantics
AUTHORED	The signer authored the document
REVIEWED	The signer reviewed the document and recommends release
APPROVED	The signer authorizes the document as a regulated record
REJECTED	The signer rejects the document

Separation of Duties (SoD) — implemented rules.

SoD flag per customer/tenant

The field `separationOfDutiesEnabled` (Boolean, default: true) is stored per customer/tenant (Prisma schema, not explicitly read out, but referenced in `signatures.ts` line 35):

```
const sodEnabled = document?.project?.system?.customer?.separationOfDutiesEnabled
 ?? true;
```

When `sodEnabled === true`, the following SoD enforcement is performed:

Implemented SoD rules

Rule 1: Author ≠ Reviewer/Approver on the same document

If a user has already signed with the meaning **AUTHORED** and attempts to sign with **REVIEWED** or **APPROVED**, an error is thrown:

```
if (['REVIEWED', 'APPROVED'].includes(data.meaning) && sameCycleMeanings.includes
('AUTHORED')) {
    throw new Error(`Cannot ${data.meaning.toLowerCase()} a document you authored
 (Separation of Duties)`);
}
```

(lines 59–61, `signatures.ts`)

Rule 2: Reviewer/Approver ≠ Author on the same document

If a user has already signed with **REVIEWED** or **APPROVED** and attempts to sign with **AUTHORED**, an error is thrown:

```
if (data.meaning === 'AUTHORED' && (sameCycleMeanings.includes('REVIEWED') || sam
eCycleMeanings.includes('APPROVED'))) {
    throw new Error('Cannot author a document you already reviewed or approved (S
eparation of Duties)');
}
```

(lines 62–64, `signatures.ts`)

SoD scope: version cycle

The SoD check is limited to the **same document version cycle**. When a document goes through a major version, previous signers can take on new roles in new cycles:

```
const currentMajor = document?.versionMajor ?? parseVersion(document?.version ??
'').major;
const sameCycleMeanings = existingSignatures
    .filter((s) => s.isValid !== false)
    .filter((s) => parseVersion(s.documentVersion ?? '').major === currentMajor)
    .map((s) => s.meaning);
```

(lines 52–57, `signatures.ts`)

Example: An author can sign as AUTHORED in version 1.0. In version 2.0, the same user may sign as REVIEWED or APPROVED.

SignatureWorkflow — sequential or parallel workflows.

Data structure

The table `SignatureWorkflow` (Prisma schema, lines 625–637) manages signature workflows:

Field	Type	Description
id	String UUID	Workflow ID
documentId	String UUID (FK)	Document
name	String	Workflow name
status	String	PENDING	IN_PROGRESS	COMPLETED	CANCELLED
workflowType	String	SEQUENTIAL	PARALLEL (default: SEQUENTIAL)
createdById	String UUID (FK)	Workflow creator
steps	SignatureStep[]	List of signature steps

SignatureStep — individual signature step

The table `SignatureStep` (Prisma schema, lines 639–653) stores each step:

Field	Type	Description
id	String UUID	Step ID
workflowId	String UUID (FK)	Association with a workflow
stepOrder	Int	Order within the workflow
signerId	String UUID (FK)	Signer ID
roleRequired	String?	Optional: required role
meaning	String	Required signature meaning
status	String	PENDING	SIGNED	REJECTED	SKIPPED

Field	Type	Description
dueDate	DateTime?	Due date
completedAt	DateTime?	When was it signed?
signature	ElectronicSignature?	Reference to the actual signature

Workflow completion logic

When a step is signed (lines 78–100, `signatures.ts`):

01 The SignatureStep is updated with `status: 'SIGNED'` and `completedAt: now()`

02 The workflow is checked: if all steps are SIGNED or SKIPPED → `status: 'COMPLETED'`

03 Otherwise the workflow remains `status: 'IN_PROGRESS'`

```
const allSigned = step.workflow.steps.every(s => s.status === 'SIGNED' || s.statu
s === 'SKIPPED');
if (allSigned) {
    await db.signatureWorkflow.update({
        where: { id: step.workflowId },
        data: { status: 'COMPLETED' },
    });
}
```

(lines 91–96, `signatures.ts`)

Server Action: signDocument.

Function

```
export async function signDocument(data: {
  documentId: string;
  password: string;
  meaning: string;
  comments?: string;
  workflowStepId?: string;
})
```

(lines 8–14, `signatures.ts`)

Flow

01 **Auth check & re-authentication**: Verify that a user is authenticated; step-up re-authentication (MFAGrant, single-use, 5-minute window) on every signature

02 **SoD enforcement**: If `separationOfDutiesEnabled`, check the SoD rules (see above)

03 **Create signature**: Call `createSignature()` (library function)

04 **Workflow update** (if `workflowStepId` is provided): - Set the SignatureStep to SIGNED/completedAt - Check all steps; set the workflow to COMPLETED if all are SIGNED/SKIPPED

Error handling

- **Unauthorized**: Throws an error if no user is authenticated
- **SoD violation**: Throws an error with a rationale (e.g. "Cannot APPROVED a document you authored")

07 — COMPONENTS

Components.

ApproverPickerDialog

Component `components/documents/ApproverPickerDialog.tsx` – selection of an approver before signature.

AuthorPickerDialog

Component `components/documents/AuthorPickerDialog.tsx` – selection of an author.

ChangeApprovalFlow

Component `components/change/ChangeApprovalFlow.tsx` – display of the approvals of a change.

08 — CODE REFERENCES

Code references.

Primary files:

- `app/actions/signatures.ts` – Server action signDocument (core logic)
- `lib/signature-service.ts` – Library functions createSignature, verifySignature
- `prisma/schema.prisma` (lines 596–623) – ElectronicSignature model
- `prisma/schema.prisma` (lines 625–637) – SignatureWorkflow model
- `prisma/schema.prisma` (lines 639–653) – SignatureStep model
- `components/documents/ApproverPickerDialog.tsx`
- `components/documents/AuthorPickerDialog.tsx`
- `components/change/ChangeApprovalFlow.tsx`

Test coverage:

- `app/actions/__tests__/signatures.test.ts`
- `components/signatures/__tests__/SigningDialog.test.tsx`

Line count: 315 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 —