Every screen — features, functions, endpoints

132 screen groups · LOCAL images only (thumbs ≤320px / Desktop pack ≤480px). No oversized conversion. Full parameter tables live in VERAE-API-REFERENCE.pdf.

Key	Devices	Title
CERT-01	`paper:CERT-01`	Certificate PDF — final (redesign)
DC-01	`ios:DC-01, android:DC-A1, windows:DC-W4`	Sign in (unified Verae + DataCube)
DC-02	`ios:DC-02, android:DC-A02, macos:DC-M02`	Create account — 5-step wizard
DC-03	`ios:DC-03, android:DC-A03, macos:DC-M03`	Two-factor verification
DC-04	`ios:DC-04, android:DC-A04, macos:DC-M04`	Two-factor settings & enrolment
DC-05	`ios:DC-05, android:DC-A2`	DataCube list (home)
DC-05N	`ios:DC-05N`	New DataCube — default ops off + previous
DC-06	`ios:DC-06, android:DC-A3`	Cube detail — files with receipt states
DC-06T	`ios:DC-06T`	File tags — on + previous selectable
DC-07	`ios:DC-07`	Add file → hash → stamp
DC-07D	`ios:DC-07D, android:DC-A07D, macos:DC-M07D`	Add files p1 — Metadata right · Upload/Tag
DC-07DM	`ios:DC-07DM, android:DC-A07DM, macos:DC-M07DM`	Add metadata per file — copy last + previo
DC-08	`ios:DC-08R, android:DC-A08R, macos:DC-M08R, debian:DC-L2, web:DC-B2`	Receipt restamp timeline — expandable JSON
DC-08R	`ios:DC-08R, android:DC-A08R, macos:DC-M08R, debian:DC-L2, web:DC-B2`	Receipt restamp timeline — expandable JSON
DC-09	`ios:DC-09, android:DC-A09, macos:DC-M09`	Cube chain — internal blocks
DC-10	`ios:DC-10, android:DC-A10, macos:DC-M10`	Verify cube / file
DC-11	`ios:DC-11, android:DC-A5`	Share & revoke (Peergos)
DC-12	`ios:DC-12, android:DC-A6`	Receipts queue (waiting for final)
DC-13	`ios:DC-13, android:DC-A13, windows:DC-W3, debian:DC-L3`	Account & settings
DC-14	`ios:DC-14, android:DC-A14, macos:DC-M14`	Usage & billing + billing key
DC-15	`ios:DC-15, debian:DC-L4`	Offline & error handling
DC-16	`ios:DC-16`	Delete → tombstone confirmation
DC-1	`ios:DC-01, android:DC-A1, macos:DC-M1, windows:DC-W4, debian:DC-L1, web:DC-B1`	Sign in (unified Verae + DataCube)
DC-2	`ios:DC-05, android:DC-A2, macos:DC-M2, windows:DC-W2, debian:DC-L2, web:DC-B2`	DataCube list (home)
DC-3	`ios:DC-06, android:DC-A3, macos:DC-M3, windows:DC-W3, debian:DC-L3`	Cube detail — files with receipt states
DC-4	`ios:DC-08R, android:DC-A08R, macos:DC-M08R, windows:DC-W4, debian:DC-L4, web:DC-B2`	Receipt restamp timeline — expandable JSON
DC-5	`ios:DC-11, android:DC-A5, macos:DC-M5`	Share & revoke (Peergos)
DC-6	`ios:DC-12, android:DC-A6, macos:DC-M6`	Receipts queue (waiting for final)
DC-JL1	`ios:DC-JL1`	Expandable JSONL data builder + previous t
DC-7	`macos:DC-M7, windows:DC-W2`	Menu bar extra + notifications
DC-8	`macos:DC-M8`	Offline & error states (Mac)
IT-01	`android:IT-01`	Device enrollment & shift sign-in
IT-02	`ios:IT-I1, android:IT-02`	Field home — iOS
IT-03	`ios:IT-I2, android:IT-03`	Scanner — iOS (VisionKit)
IT-04	`android:IT-04`	RFID bulk read (cage count)
IT-05	`android:IT-05`	Pickup check-in (GPS GeoFix)
IT-06	`android:IT-06`	Item capture — photos & identity
IT-07	`android:IT-07`	Custody hand-off — QR exchange
IT-08	`ios:IT-I3, android:IT-08`	Signature — iOS (PencilKit)
IT-09	`ios:IT-I4, android:IT-09`	Grading — iOS

Key	Devices	Title
IT-10	`android:IT-10`	Floor mode — shred bin (and wipe check)
IT-11	`android:IT-11`	Offline queue
IT-12	`android:IT-12`	Glove / outdoor (sunlight) mode
IT-13	`android:IT-13`	Lot receipts & custody timeline
IT-C1	`web:IT-C1`	Customer portal — lots
IT-C2	`web:IT-C2`	Customer portal — lot detail
IT-C3	`web:IT-C7`	Item history — mobile web
IT-C4	`web:IT-C4`	Certificate view + verify
IT-C5	`web:IT-C5`	Request pickup & quote
IT-C6	`web:IT-C6`	Settlement statements
IT-C7	`web:IT-C7`	Item history — mobile web
IT-I1	`ios:IT-I1, android:IT-02`	Field home — iOS
IT-I2	`ios:IT-I2, android:IT-03`	Scanner — iOS (VisionKit)
IT-I3	`ios:IT-I3, android:IT-08`	Signature — iOS (PencilKit)
IT-I4	`ios:IT-I4, android:IT-09`	Grading — iOS
IT-P1	`web:IT-P1`	Partner — shared lots
IT-P2	`web:IT-P2`	Partner — accept custody hand-off
IT-P3	`web:IT-P3`	Partner — upload evidence
IT-P4	`web:IT-P4`	Partner — API keys & webhooks
IT-S1	`windows:IT-S1`	Station kiosk sign-in (badge + PIN)
IT-S2	`windows:IT-S2`	Dock receipt (shipment level)
IT-S3	`windows:IT-S3, debian:IT-SL2`	Receiving reconciliation vs manifest
IT-S4	`macos:IT-SM1, windows:IT-S4`	Supervisor approvals (Mac)
IT-S5	`windows:IT-S5, debian:IT-SL1`	Wipe station (per-drive bays)
IT-S6	`windows:IT-S6`	Shred / destruction station
IT-S7	`windows:IT-S7, debian:IT-SL3`	Station offline (leaf buffering)
IT-SL1	`windows:IT-S5, debian:IT-SL1`	Wipe station (per-drive bays)
IT-SL2	`windows:IT-S3, debian:IT-SL2`	Receiving reconciliation vs manifest
IT-SL3	`windows:IT-S7, debian:IT-SL3`	Station offline (leaf buffering)
IT-SM1	`macos:IT-SM1, windows:IT-S4`	Supervisor approvals (Mac)
IT-W1	`web:IT-W1`	Ops console — floor & dock dashboard
IT-W2	`web:IT-W2`	Lot detail — timeline & evidence
IT-W3	`web:IT-W3`	Pending-receipt queue & chain status
IT-W4	`web:IT-W4`	Pickups — schedule & assign trucks
IT-W5	`web:IT-W5`	Exceptions & cases
IT-W6	`web:IT-W6`	Certificate lifecycle
MT-M1	`macos:MT-M1`	Verae Timestamping (Mac) — drop, hash, rec
MT-M2	`macos:MT-M2`	Export sheet with not-final guard
PT-01	`macos:PT-01, windows:PT-W1, debian:PT-L1`	Print dialog → “Verae Timestamping” printe
PT-02	`macos:PT-02, windows:PT-W2, debian:PT-L2`	Sign-in prompt — choose how to connect

Key	Devices	Title
PT-03	`macos:PT-03`	Sign in · 2FA · password reset
PT-04	`macos:PT-04, windows:PT-W4, debian:PT-L3`	Print job queue + receipt status
PT-05	`macos:PT-05`	Final receipt + certificate ready
PT-06	`macos:PT-06, windows:PT-W3`	Menu bar status + notifications (Mac)
PT-07	`macos:PT-07`	Settings (Mac)
PT-08	`macos:PT-08`	Offline & server-error states (printit)
PT-A1	`android:PT-A1`	Android PrintService “Verae Timestamping”
PT-B1	`web:PT-B1`	printit.verae.ai — download & how it works
PT-I1	`ios:PT-I1`	iOS share extension “Timestamp with Verae”
PT-L1	`macos:PT-01, windows:PT-W1, debian:PT-L1`	Print dialog → “Verae Timestamping” printe
PT-L2	`macos:PT-02, windows:PT-W2, debian:PT-L2`	Sign-in prompt — choose how to connect
PT-L3	`macos:PT-04, windows:PT-W4, debian:PT-L3`	Print job queue + receipt status
PT-W1	`macos:PT-01, windows:PT-W1, debian:PT-L1`	Print dialog → “Verae Timestamping” printe
PT-W2	`macos:PT-02, windows:PT-W2, debian:PT-L2`	Sign-in prompt — choose how to connect
PT-W3	`macos:PT-06, windows:PT-W3`	Menu bar status + notifications (Mac)
PT-W4	`macos:PT-04, windows:PT-W4, debian:PT-L3`	Print job queue + receipt status
SG-A1	`ios:SG-I1, android:SG-A1`	iOS — signature request + sign
SG-I1	`ios:SG-I1, android:SG-A1`	iOS — signature request + sign
SG-I2	`ios:SG-I2, macos:SG-M1, windows:SG-WN1, web:SG-W1`	iOS — send for signature (from a cube file
SG-L1	`debian:SG-L1, web:SG-W5`	Debian GNOME — completion notification + S
SG-M1	`ios:SG-I2, macos:SG-M1, windows:SG-WN1, web:SG-W1`	iOS — send for signature (from a cube file
SG-P1	`paper:SG-P1`	Signature audit certificate (PDF)
SG-W1	`ios:SG-I2, macos:SG-M1, windows:SG-WN1, web:SG-W1`	iOS — send for signature (from a cube file
SG-W2	`web:SG-W2`	Request status — step stamps
SG-W3	`web:SG-W3`	Signer landing + email OTP
SG-W4	`web:SG-W4`	Sign — draw / type / upload + consent
SG-W5	`debian:SG-L1, web:SG-W5`	Debian GNOME — completion notification + S
SG-WN1	`ios:SG-I2, macos:SG-M1, windows:SG-WN1, web:SG-W1`	iOS — send for signature (from a cube file
ZP-01	`web:ZP-01`	Portal sign-in + 2FA
ZP-02	`web:ZP-09`	Portal — mobile responsive
ZP-03	`web:ZP-03`	Usage & metering detail
ZP-04	`web:ZP-04`	Invoices
ZP-05	`web:ZP-05`	Statement (ledger)
ZP-06	`web:ZP-06`	Billing — balance, reload, cards, PO
ZP-07	`web:ZP-07`	Security — API keys (xVerae-) & 2FA
ZP-08	`web:ZP-08`	API & pricing
ZP-09	`web:ZP-09`	Portal — mobile responsive
ZP-B0	`web:ZP-B0`	Staff sign-in (Verae IAM)
ZP-B1	`web:ZP-B1`	Admin — customers
ZP-B2	`web:ZP-B2`	Customer support lookup

Key	Devices	Title
ZP-B3	`web:ZP-B3`	Sales — pipeline & quotes
ZP-B4	`web:ZP-B4`	Admin — rate card & customer types
ZP-B5	`web:ZP-B5`	Finance — invoices, reconciliation, period
ZP-B6	`web:ZP-B6`	Finance — reports
ZP-B7	`web:ZP-B7`	Compliance — audit log, evidence packs, ch
ZP-B8	`web:ZP-B8`	Admin — system health & users
ZP-B9	`web:ZP-B9`	Staff IAM — people, roles & interface swit
ZP-V1	`web:ZP-V1`	Verify — entry (file, hash, QR)
ZP-V2	`web:ZP-V5`	Verify — mobile (QR scan landing)
ZP-V3	`web:ZP-V5`	Verify — mobile (QR scan landing)
ZP-V4	`web:ZP-V4`	Verify — not found / tampered / revoked
ZP-V5	`web:ZP-V5`	Verify — mobile (QR scan landing)

CERT-01 — Certificate PDF — final (redesign)

The printit certificate redesigned around the final-receipt rule: status box, multihash, server time, Substrate record, proof and a QR/verify link.

paper

CERT-01

Features	
• Brand + certificate number + verify domain.	• Status box: 'FINAL — Substrate block #N, finalized' with UTC and ET.
• Fields: multihash and hex, server time (never local), VeraeTime receipt (CIDv1, not Qm…), Substrate block/extrinsic/hash, proof location.	• QR + human-readable verify link + 3-step how-to.
• Digest-only statement and multihash definition.	

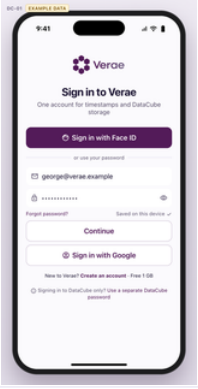
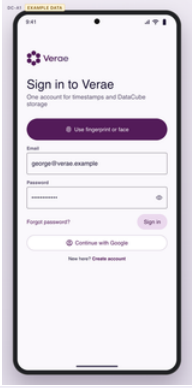
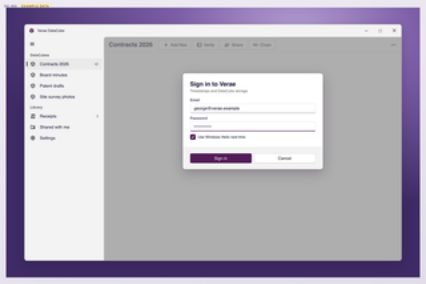
Function / source references
`PrintToVeraeTimeStamp/internal/pdfcert/pdfcert.go`
`platform/registration-pdf/build.mjs`
`site/printit.verae.ai/samples/verae-receipt-sample.pdf`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-01 — Sign in (unified Verae + DataCube)

One sign-in for the Verae account (zap.verae.ai, timestamps, billing) and the Peergos DataCube account, resolving finding U-i10 (two accounts confuse users).

ios	android	windows
		
DC-01	DC-A1	DC-W4

Features	
• Brand header with the Verae mark; plain-language promise of what the account covers.	• Primary action is biometric when saved logins exist (Keychain on iOS; KeyStore strong biometric on Android).
• Email + password fallback; password field shows focus ring (WCAG 2.4.7) and a reveal toggle with an accessible label.	• Google sign-in kept from social_login.dart.
• Create account entry; Free plan copy taken from verae_first_login.dart.	• Advanced path for DataCube-only sign-in kept but de-emphasised.
• BiometricPrompt (BIOMETRIC_STRONG, KeyStore-backed); declare USE_BIOMETRIC (U-a2 verify).	• Outlined text fields with floating labels (M3).
• Credential Manager for Google / passkeys.	• ContentDialog with primary/close buttons in footer; Enter/Esc.

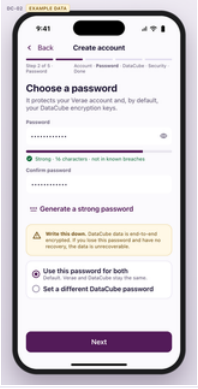
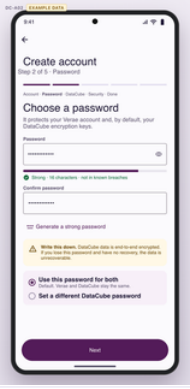
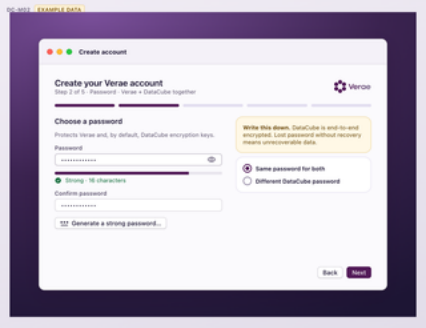
Function / source references
`lib/main.dart LoginScreen`
`lib/auth/verae_first_login.dart VeraeLoginScreen`
`auth/biometric_login.dart`
`auth/social_login.dart`
`auth/saved_logins.dart`
`lib/main.dart`
`lib/auth/biometric_login.dart (KeyStore strong biometric)`
`lib/auth/verae_first_login.dart`
`lib/auth/biometric_login.dart`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-02 — Create account — 5-step wizard

Sign-up that creates the Verae account and the DataCube (Peergos) account together, with free movement between steps.

ios	android	macos
		
DC-02	DC-A02	DC-M02

Features	
• Stepper with named steps; steps are tappable to move freely (commit 32c7011).	• Password with focus ring and reveal.
• Strength meter uses text + icon, never colour alone (WCAG 1.4.1).	• Generate password (auth/generated_password.dart).
• Unrecoverable-data warning copied from verae_first_login.dart, given a warning container.	• Same-password choice (commits 35d2342, 3c9774e).
• Linear 5-step progress.	• Password fields + strength text+icon.
• Generate password.	• Unrecoverable-data warning.
• Same vs different DataCube password.	• 5-step bar.

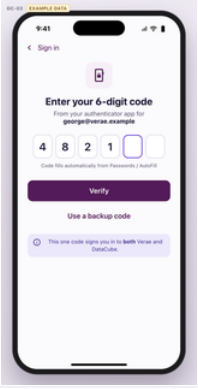
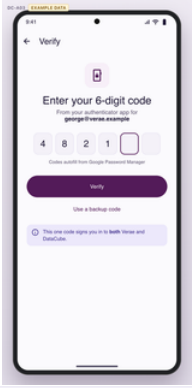
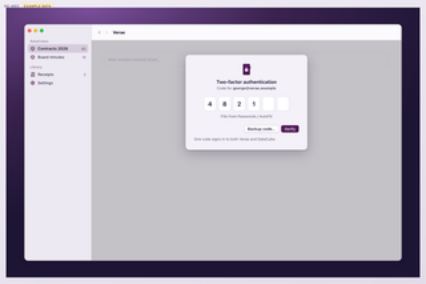
Function / source references
`lib/signup_screen.dart CreateAccountScreen, VeraeAccountStep`
`lib/auth/verae_first_login.dart (Step n of 5)`
`lib/signup_rules.dart`
`lib/datacube/datacube_ui.dart`
`lib/auth/*`
`lib/billing/*`
`lib/signup_screen.dart`

API id	Method	Path	Key params (name:type)
`portal.signup`	**POST**	`customer.verae.ai/portal/api/signup`	name:string, email:string (email), password:string (≥8), acceptTos:boolean
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-03 — Two-factor verification

Single TOTP step shared by Verae and DataCube sign-in (today TOTP appears in two places: main sign-in and the timestamps portal dialog).

ios	android	macos
		
DC-03	DC-A03	DC-M03

Features	
• Six segmented digit boxes; one-time-code content type so AutoFill can fill it.	• AutoFill hint.
• Verify; auto-submits on the 6th digit.	• Backup codes path (two_factor_screen.dart backup-codes).
• Explains a single code covers both accounts.	• Six digit cells + autofill.
• Verify + backup code.	• Both-accounts banner.
• TOTP digits.	• Backup + Verify.
• Both-accounts note.	

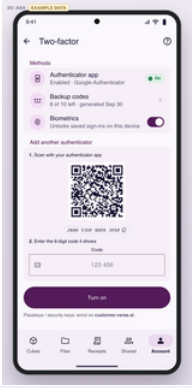
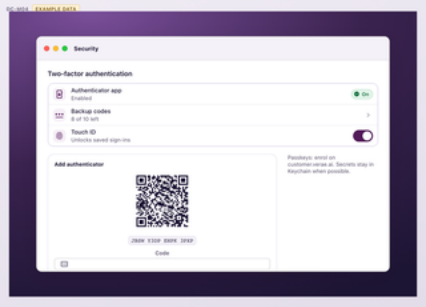
Function / source references
`lib/auth/two_factor_screen.dart`
`lib/timestamps/timestamps_ui.dart _VtLoginDialog (TOTP)`
`lib/auth/verae_first_login.dart`
`lib/datacube/datacube_ui.dart`
`lib/auth/*`
`lib/billing/*`
`lib/signup_screen.dart`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`portal.2fa`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/2fa/*`	setup:POST {}, code:string, totpCode:string

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-04 — Two-factor settings & enrolment

Manage authenticator, backup codes and biometric unlock; enrol a new authenticator with QR + manual secret.

ios	android	macos
		
DC-04	DC-A04	DC-M04

Features	
• QR to otpauth:// URI (issuer Verae); VoiceOver label 'QR code for authenticator setup'.	• Manual secret, grouped in 4s, copy button.
• Turn on is enabled after a valid code.	• WebAuthn note from current code.
• Methods list.	• QR + secret + turn on.
• Portal note for passkeys.	• Portal note.

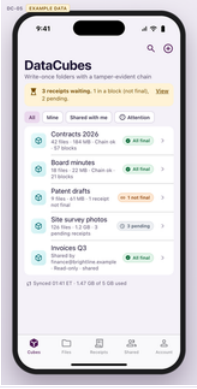
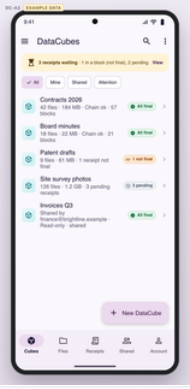
Function / source references
`lib/auth/two_factor_screen.dart (_EnrollDialog, backup codes)`
`lib/auth/authenticator_setup.dart`
`lib/datacube/datacube_ui.dart`
`lib/auth/*`
`lib/billing/*`
`lib/signup_screen.dart`

API id	Method	Path	Key params (name:type)
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`portal.2fa`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/2fa/*`	setup:POST {}, code:string, totpCode:string

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-05 — DataCube list (home)

Home screen: all cubes with an at-a-glance evidence status per cube and a global 'receipts waiting' indicator.

ios	android
	
DC-05	DC-A2

Features	
• Global pending banner (U-i3): count of not-final receipts with a jump to the Receipts queue.	• Filter chips incl. 'Attention'.
• Cube rows: cube glyph, size, chain health, status badge with text (never colour only).	• Sync + storage line (billing context).
• Tonal warning card for pending receipts.	• M3 filter chips with check icon when selected.
• Extended FAB 'New DataCube' (collapses on scroll); navigation bar with 5 destinations; predictive back (U-a3).	

Function / source references
`lib/datacube/datacube_ui.dart CubeListScreen, DataCubesEdgeTab (empty text 'No DataCubes yet.')`
`lib/datacube/datacube_ui.dart CubeListScreen`
`file_browser.dart bottom bar`

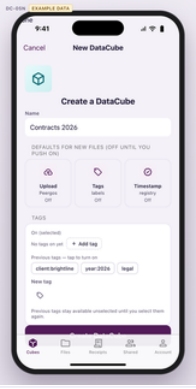
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-05N — New DataCube — default ops off + previous tags

Create cube; default Upload/Tags/Timestamp icon toggles start off; previous tags library.

ios



DC-05N

Features	
• Name.	• Three ops off.
• Previous tags.	• Create.

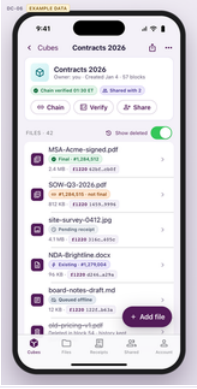
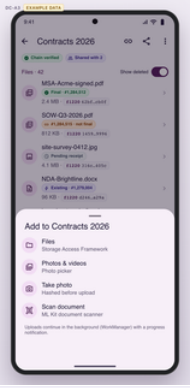
Function / source references
`lib/files/*`
`lib/datacube/*`
`lib/timestamps/*`
`peergos`
`customer.verae.ai/v1/timestamp`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`portal.signup`	**POST**	`customer.verae.ai/portal/api/signup`	name:string, email:string (email), password:string (≥8), acceptTos:boolean
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-06 — Cube detail — files with receipt states

Shows a cube's files, each with its receipt state per George's rule, plus tombstoned (deleted) entries kept in history.

ios	android
	
DC-06	DC-A3

Features	
• Cube header: owner, block count, chain verification time, share count.	• Primary cube actions: Chain (DC-09), Verify (DC-10), Share (DC-11).
• 'Show deleted' toggle (CheckedPopupMenuItem 'Show deleted' today).	• File rows: name, size, short multihash, receipt badge: Final · #N / #N · not final / Pending receipt / Existing / Queued offline. Tombstones struck through with 'Deleted in block 54'.
• Add file → DC-07 (Files, Photos, Camera, Scan).	• Same receipt badges as iOS (shared component).
• M3 modal bottom sheet replaces the iOS action sheet.	• WorkManager for queued uploads and receipt reconcile (U-a5) with a foreground progress notification.

Function / source references
`lib/datacube/datacube_ui.dart CubeScreen (tombstoned rows, 'Show deleted', 'Timestamp un-stamped files')`
`lib/datacube/datacube_ui.dart CubeScreen`
`files/sync_queue.dart`
`lib/files/file_browser.dart`
`lib/timestamps/auto_stamp.dart`

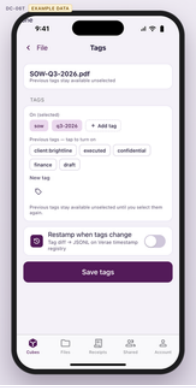
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-06T — File tags — on + previous selectable

Edit tags; previous tags unselected until tapped; optional restamp default off.

ios



DC-06T

Features	
• On tags.	• Previous tags.
• Restamp toggle off.	• Save.

Function / source references
`lib/files/*`
`lib/datacube/*`
`lib/timestamps/*`
`peergos`
`customer.verae.ai/v1/timestamp`
`lib/files/file_browser.dart`
`lib/timestamps/auto_stamp.dart`

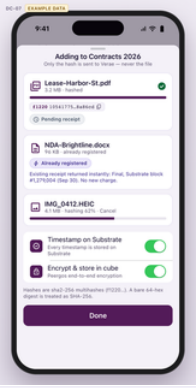
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-07 — Add file → hash → stamp

Adds files to a cube: hashes on device, checks if already registered (instant existing receipt), stamps new hashes, encrypts and stores in the cube.

ios



DC-07

Features	
• Verae timestamp registry (hash only)	• Expandable nested JSONL builder + path diffs
• Copy last metadata per item	• Previous tags unselected — tap to turn on
• Upload / Tags / Timestamp: separate icon toggles, DEFAULT OFF	• Metadata button on the RIGHT of each file row
• Per-file card: hashing progress → multihash → 'Pending receipt'.	• Duplicate path: hash already registered → 'Already registered' badge and the existing final receipt immediately (doc 21 rule).
• Large-file hashing in the Rust core, with progress; cancel per file.	• Stamp and encrypt toggles; stamping is on by default and always goes to Substrate.
• Multihash rule stated plainly.	

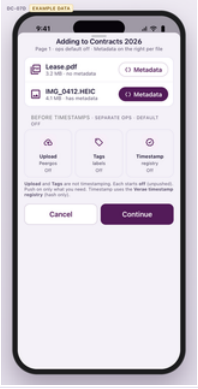
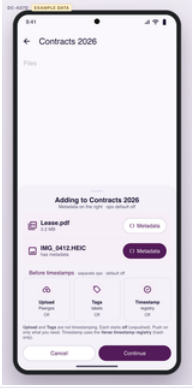
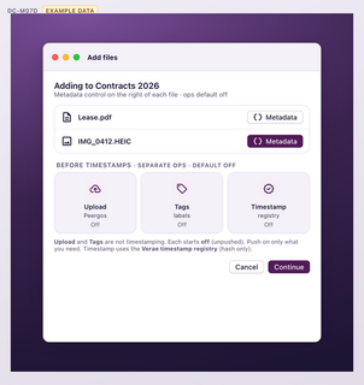
Function / source references
`lib/datacube/datacube_ui.dart (Add files / photos dialogs, 'Adding to DataCube…')`
`lib/files/file_browser.dart (image_picker camera)`
`lib/timestamps/auto_stamp.dart`
`lib/files/file_browser.dart`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-07D — Add files p1 — Metadata right · Upload/Tags/Timestamp off

Page 1 Adding to Contracts: per-file Metadata button on the right; Upload, Tags, Timestamp as separate icon toggles default off. Metadata sheet: copy last metadata, previous tags selectable, JSONL bui

ios	android	macos
		
DC-07D	DC-A07D	DC-M07D

Features	
• Verae timestamp registry (hash only)	• Expandable nested JSONL builder + path diffs
• Copy last metadata per item	• Previous tags unselected — tap to turn on
• Upload / Tags / Timestamp: separate icon toggles, DEFAULT OFF	• Metadata button on the RIGHT of each file row
• File rows with Metadata button on the right.	• Upload / Tags / Timestamp icon toggles — default Off (unpushed).
• Upload ≠ Tags ≠ Timestamp.	• Continue / Cancel.
• Metadata right.	• Upload/Tags/Timestamp off.

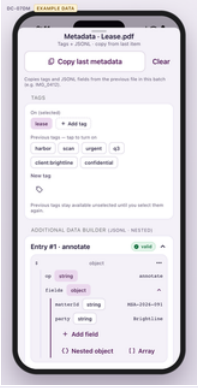
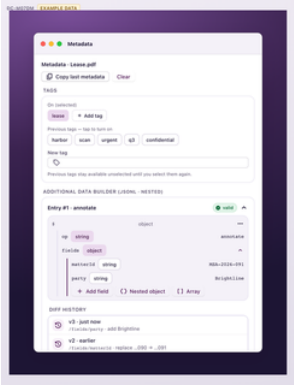
Function / source references
`lib/files/*`
`lib/datacube/*`
`lib/timestamps/*`
`peergos`
`customer.verae.ai/v1/timestamp`
`lib/files/file_browser.dart`
`lib/timestamps/auto_stamp.dart`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-07DM — Add metadata per file — copy last + previous tags

Per-item metadata: Copy last metadata; tags on + previous unselected tags to turn on; expandable JSONL builder.

ios	android	macos
		
DC-07DM	DC-A07DM	DC-M07DM

Features	
• Verae timestamp registry (hash only)	• Expandable nested JSONL builder + path diffs
• Copy last metadata per item	• Previous tags unselected — tap to turn on
• Upload / Tags / Timestamp: separate icon toggles, DEFAULT OFF	• Metadata button on the RIGHT of each file row
• Copy last metadata.	• Tags on + previous tags (tap to select).
• JSONL nested builder + diff history.	• Save metadata.
• Copy last.	• Previous tags.

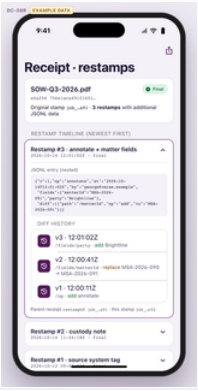
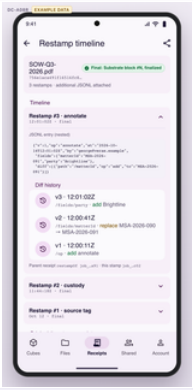
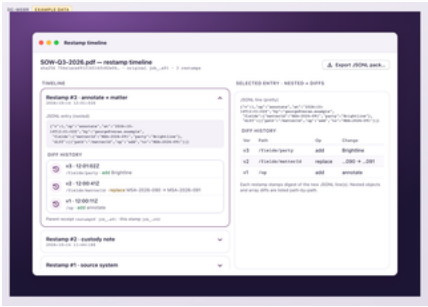
Function / source references
`lib/files/*`
`lib/datacube/*`
`lib/timestamps/*`
`peergos`
`customer.verae.ai/v1/timestamp`
`lib/files/file_browser.dart`
`lib/timestamps/auto_stamp.dart`

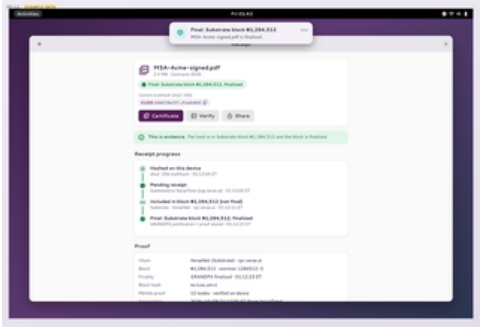
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-08 — Receipt restamp timeline — expandable JSONL + diffs

Shows a file's original timestamp receipt and each restamp with additional JSONL payloads. Entries expand to nested JSON and per-field diff history.

ios	android	macos
		
DC-08R	DC-A08R	DC-M08R

debian	web
	
DC-L2	DC-B2

Features	
• Header: file, sha256, restamp count.	• Timeline newest-first; expand/collapse.
• Expanded: pretty JSONL nested object + diff history list (path, op, from/to).	• restampOf link to parent job.
• Original receipt footer (no additional data).	• Contract note: restamp:true; digests of additional JSONL.
• M3 top app bar.	• Expandable cards.
• JSONL + diff list.	• Original footer.
• Split: timeline | detail.	• JSONL pretty + diff table.

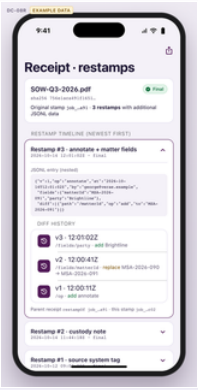
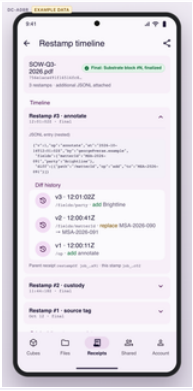
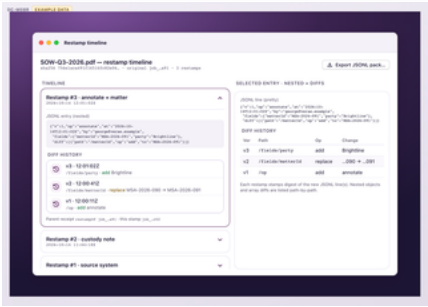
Function / source references
`lib/timestamps/*`
`CORE-CONTRACT restampOf`
`archive JSONL additional`
`lib/timestamps/timestamps_ui.dart`
`New`

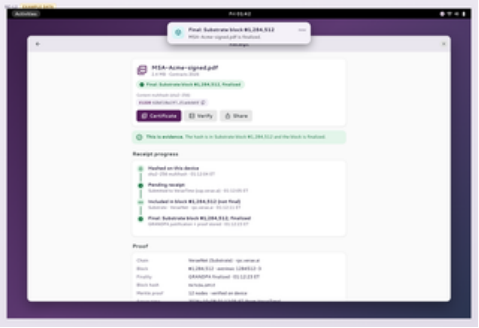
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-08R — Receipt restamp timeline — expandable JSONL + diffs

Shows a file's original timestamp receipt and each restamp with additional JSONL payloads. Entries expand to nested JSON and per-field diff history.

ios	android	macos
		
DC-08R	DC-A08R	DC-M08R

debian	web
	
DC-L2	DC-B2

Features	
• Header: file, sha256, restamp count.	• Timeline newest-first; expand/collapse.
• Expanded: pretty JSONL nested object + diff history list (path, op, from/to).	• restampOf link to parent job.
• Original receipt footer (no additional data).	• Contract note: restamp:true; digests of additional JSONL.
• M3 top app bar.	• Expandable cards.
• JSONL + diff list.	• Original footer.
• Split: timeline | detail.	• JSONL pretty + diff table.

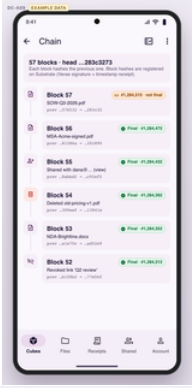
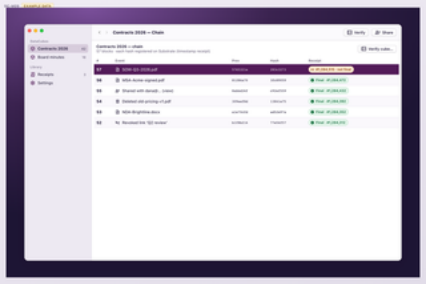
Function / source references
`lib/timestamps/*`
`CORE-CONTRACT restampOf`
`archive JSONL additional`
`lib/timestamps/timestamps_ui.dart`
`New`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-09 — Cube chain — internal blocks

Visualises the cube's internal hash chain: adds, shares, revokes and tombstones as blocks, each with its own Substrate registration state.

ios	android	macos
		
DC-09	DC-A09	DC-M09

Features	
• Chain summary + head hash; explains linkage in one sentence.	• Block cards: sequence, event (add / share / revoke / tombstone), prev → this hash, per-block receipt state. Tombstone blocks are red-tinted; deletes never rewrite history.
• Summary + head hash.	• Block cards with event type and receipt stage.
• Summary.	• Table seq/event/prev/hash/receipt.
• Verify → DC-M10.	

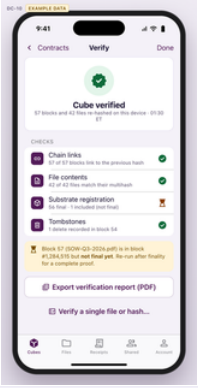
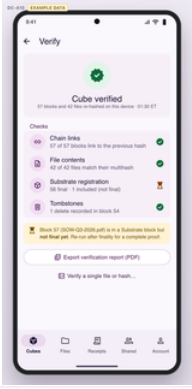
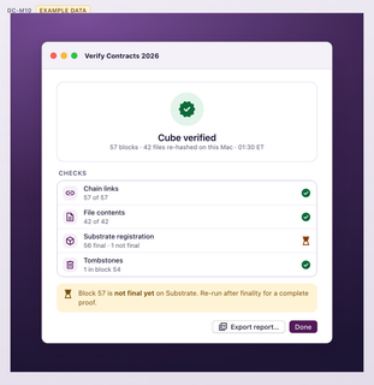
Function / source references
`lib/datacube/datacube_ui.dart CubeChainScreen ('Chain (all blocks)', seq · blockId · createdAt · sha256)`
`lib/datacube/datacube_ui.dart`
`lib/auth/*`
`lib/billing/*`
`lib/signup_screen.dart`
`CubeChainScreen`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-10 — Verify cube / file

Re-hashes all blocks and files locally and checks Substrate registrations; reports success, not-final items and tampering.

ios	android	macos
		
DC-10	DC-A10	DC-M10

Features	
• Result hero with icon + words (not colour only).	• Check list: chain links, file contents, Substrate registration, tombstones.
• Not-final caveat banner.	• Exportable verification report (itself stamped).
• Result hero.	• Check list.
• Not-final banner.	• Export + single-file verify.
• Export + Done.	

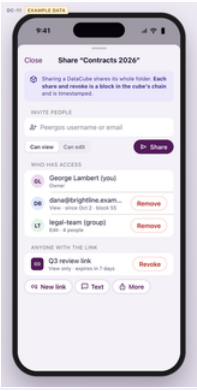
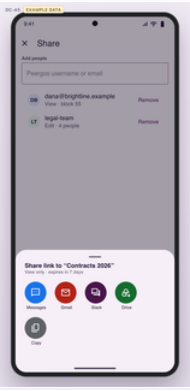
Function / source references
`lib/datacube/datacube_ui.dart CubeVerifyScreen ('Verify chain', 'Re-hashing blocks and files…')`
`lib/src/rust/api/datacube.dart`
`lib/datacube/datacube_ui.dart`
`lib/auth/*`
`lib/billing/*`
`lib/signup_screen.dart`
`CubeVerifyScreen`

API id	Method	Path	Key params (name:type)
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-11 — Share & revoke (Peergos)

Share a cube or file with Peergos users/groups or a link; revoke access. Shares and revokes are cube-chain blocks with their own timestamps.

ios	android
	
DC-11	DC-A5

Features	
• Chain-recording explanation (existing copy, elevated).	• Invite by Peergos username/email with view/edit segmented control.
• Access list with Remove (destructive style + confirmation).	• Links with expiry and Revoke.
• Channels: new link, SMS (existing 'Send by text'), iOS share sheet.	• Full-screen dialog pattern on compact width (M3).
• Android sharesheet with direct-share targets; revoke uses an M3 dialog (same copy as iOS).	

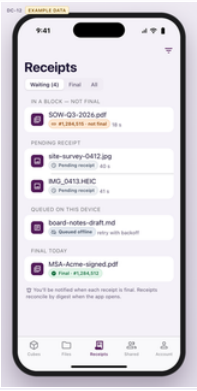
Function / source references
`lib/sharing/sharing_ui.dart ShareScreen, _NewLinkDialog, PeopleScreen, LinksScreen`
`lib/src/rust/api/sharing.dart revokeAccess`
`lib/sharing/sharing_ui.dart (share sheet)`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-12 — Receipts queue (waiting for final)

A visible queue of every hash not yet final, grouped by stage, fixing U-i3 (pending jobs hidden until 'Verify' tapped).

ios	android
	
DC-12	DC-A6

Features	
• Segment: Waiting / Final / All.	• Groups by stage — included (not final), pending, queued offline, final today.
• Offline queue items with retry/backoff from auto_stamp.dart's persistent queue.	• Push (EP-044) + reconcile-on-launch explanation.
• Offline tonal card; snackbar on reconnect with action.	

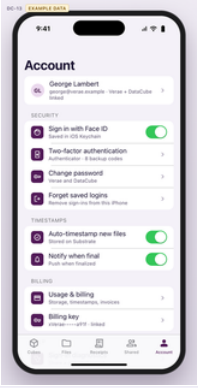
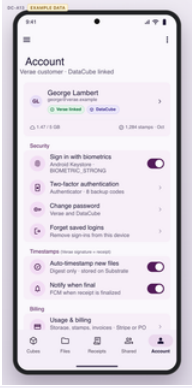
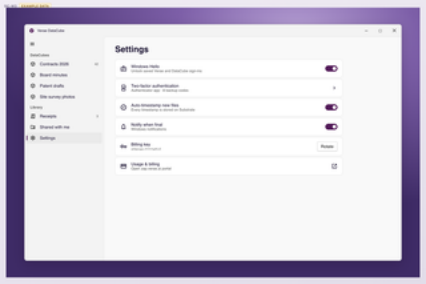
Function / source references
`lib/timestamps/timestamps_ui.dart TimestampsScreen`
`lib/timestamps/auto_stamp.dart (persistent queue, backoff)`
`lib/files/sync_queue.dart`
`lib/timestamps/auto_stamp.dart`
`lib/errors.dart`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

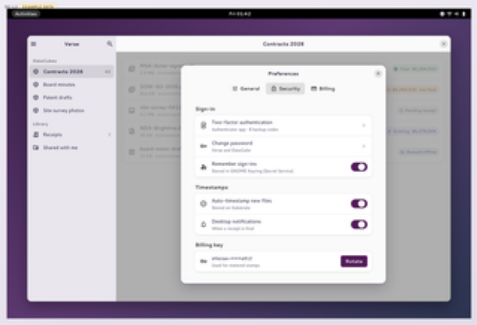
Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-13 — Account & settings

Account, security, timestamp preferences, billing key and diagnostics in grouped iOS settings style.

ios	android	windows
		
DC-13	DC-A13	DC-W3

debian



DC-L3

Features	
• Linked identity row (Verae + DataCube).	• Security group: Face ID, 2FA (DC-04), change password, forget saved logins (existing strings).
• Timestamp preferences: auto-stamp, notify on final.	• Billing: Usage & billing (DC-14) and the billing key (xVerae- token) masked.
• Profile hero: avatar, email, Verae+DataCube linked chips, storage/stamps summary; tappable to edit profile (portal).	• Security group: BiometricPrompt toggle (Keystore), 2FA → DC-04/Android equivalent, change password, forget logins.
• Timestamps group: auto-stamp + FCM notify-on-final (Verae signature = timestamp receipt).	• Billing group: Usage & billing (DC-14 companion) + masked xVerae- key; footer points to customer.verae.ai for Stripe/PO.
• Help: diagnostics + about. Bottom navigation Account selected (5 destinations).	• Settings cards; Windows Hello replaces Face ID (credentials in Credential Locker).
• AdwPreferencesDialog with pages General / Security / Billing.	• Billing key row with Rotate.

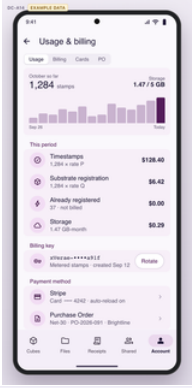
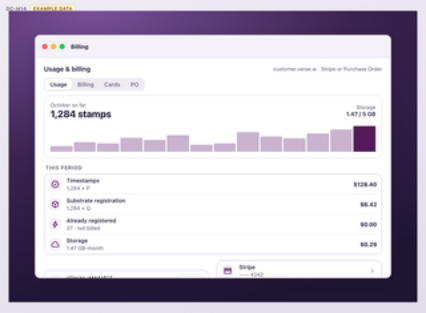
Function / source references
`lib/settings_screen.dart SettingsScreen`
`lib/diag/sign_in_diag.dart DiagPanel`
`auth/change_password.dart`
`lib/auth/biometric_login.dart`
`lib/billing/usage_billing_screen.dart`
`design DC-13`
`lib/settings_screen.dart`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.2fa`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/2fa/*`	setup:POST {}, code:string, totpCode:string

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-14 — Usage & billing + billing key

Usage, charges and the billing key in-app; links to the zap portal for invoices and payment.

ios	android	macos
		
DC-14	DC-A14	DC-M14

Features	
• Tabs kept from current app: Usage / Billing / Cards / PO (PO only in staff mode).	• Period summary with daily chart; chart has a text alternative.
• Line items as units × price variables (P, Q); already-registered hits shown as not billed.	• Billing key (xVerae-) masked with rotate.
• Price-variable disclaimer + portal link.	• Tabs Usage/Billing/Cards/PO.
• Chart + line items.	• Billing key.
• Stripe + PO.	• customer.verae.ai disclaimer.
• Tabs.	• Key rotate.

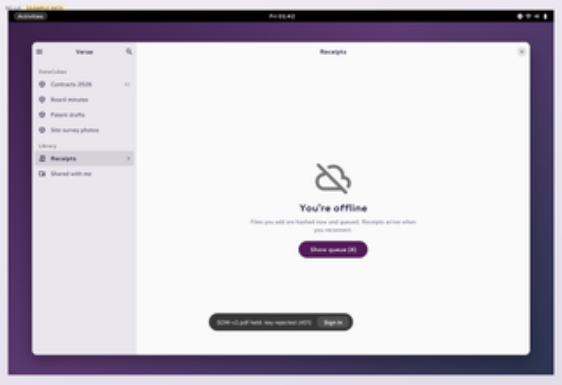
Function / source references
`lib/billing/usage_billing_screen.dart (_UsageTab, _BillingTab, _CardsTab, _PoTab, _Banner 'Showing sample data…')`
`lib/billing/billing_api.dart`
`lib/src/rust/api/billing_link.dart`
`lib/datacube/datacube_ui.dart`
`lib/auth/*`
`lib/billing/*`
`lib/signup_screen.dart`
`lib/billing/usage_billing_screen.dart`

API id	Method	Path	Key params (name:type)
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-15 — Offline & error handling

How the app behaves offline and on hard errors: hashes are captured and queued, nothing is lost, and actions needing the user are explicit.

ios	debian
	
DC-15	DC-L4

Features	
• Persistent offline banner (not just a snackbar).	• Queued items show their computed multihash so the user knows capture happened.
• 'Action needed' errors (401/403) held, not dropped — mirrors printit's pending folder.	• AdwStatusPage with pill button.
• AdwToast for 'action needed'.	

Function / source references
`lib/errors.dart isOfflineError`
`lib/files/sync_queue.dart`
`lib/files/transfers.dart`
`lib/timestamps/auto_stamp.dart`
`lib/errors.dart`

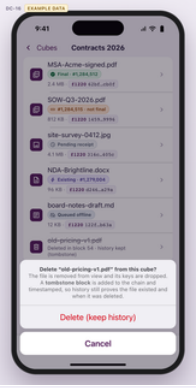
API id	Method	Path	Key params (name:type)
`peergos.cube`	—	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-16 — Delete → tombstone confirmation

Deletes in a DataCube are tombstones: the file disappears but a timestamped block records the delete. The action sheet explains this before confirming.

ios



DC-16

Features	
• Action sheet copy explains tombstone semantics in plain words.	• Destructive button wording 'Delete (keep history)'.

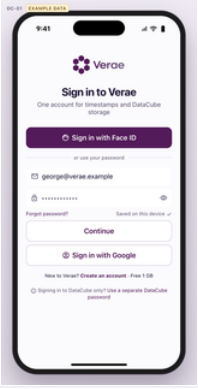
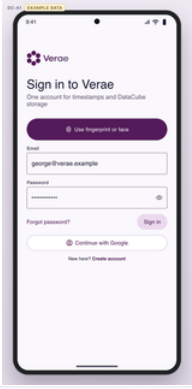
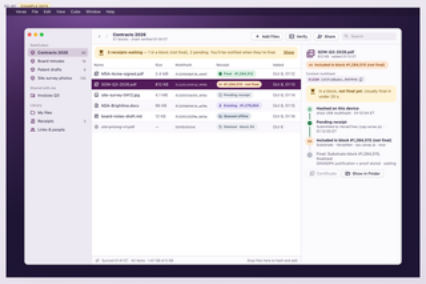
Function / source references
`lib/datacube/datacube_ui.dart (o.tombstoned, tombstoneBlock, 'deleted in …')`
`lib/files/file_browser.dart delete dialog`

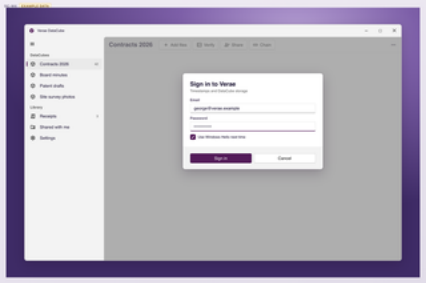
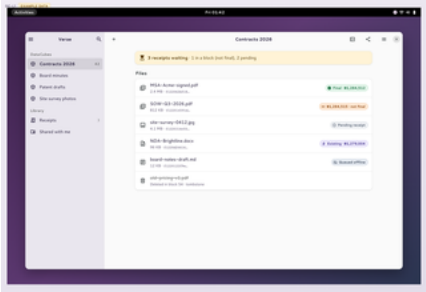
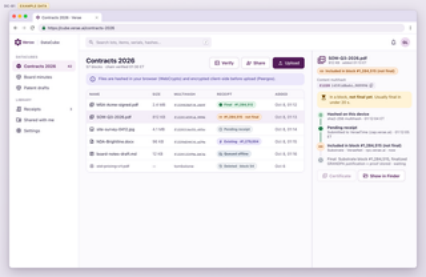
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-1 — Sign in (unified Verae + DataCube)

One sign-in for the Verae account (zap.verae.ai, timestamps, billing) and the Peergos DataCube account, resolving finding U-i10 (two accounts confuse users).

ios	android	macos
		
DC-01	DC-A1	DC-M1

windows	debian	web
		
DC-W4	DC-L1	DC-B1

Features	
• Brand header with the Verae mark; plain-language promise of what the account covers.	• Primary action is biometric when saved logins exist (Keychain on iOS; KeyStore strong biometric on Android).
• Email + password fallback; password field shows focus ring (WCAG 2.4.7) and a reveal toggle with an accessible label.	• Google sign-in kept from social_login.dart.
• Create account entry; Free plan copy taken from verae_first_login.dart.	• Advanced path for DataCube-only sign-in kept but de-emphasised.
• BiometricPrompt (BIOMETRIC_STRONG, KeyStore-backed); declare USE_BIOMETRIC (U-a2 verify).	• Outlined text fields with floating labels (M3).
• Credential Manager for Google / passkeys.	• Sidebar (source list): cubes with counts, Shared with me, Library (Receipts count = waiting). Replaces the Material drawer (U-w5).
• Waiting-receipts banner.	• Sortable table: name, size, short multihash, receipt badge, added time; tombstones struck through.

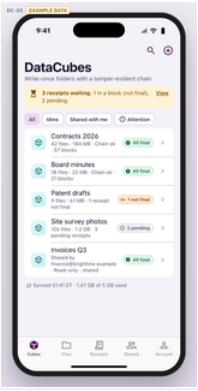
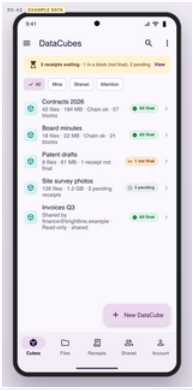
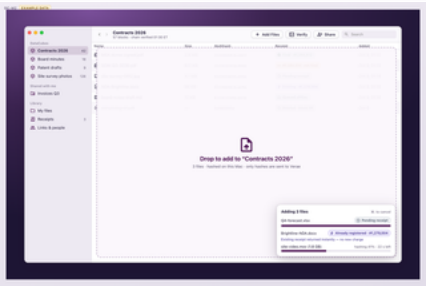
Function / source references
`lib/main.dart LoginScreen`
`lib/auth/verae_first_login.dart VeraeLoginScreen`
`auth/biometric_login.dart`
`auth/social_login.dart`
`auth/saved_logins.dart`
`lib/main.dart`
`lib/auth/biometric_login.dart (KeyStore strong biometric)`
`lib/datacube/datacube_ui.dart`
`lib/files/file_browser.dart (drawer, desktop_drop, 'Save to computer')`
`lib/files/item_detail.dart`

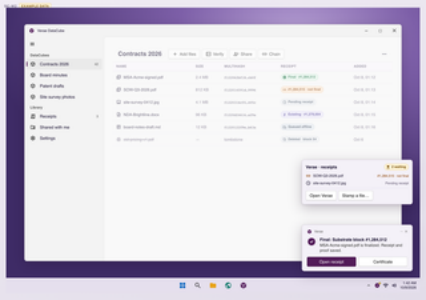
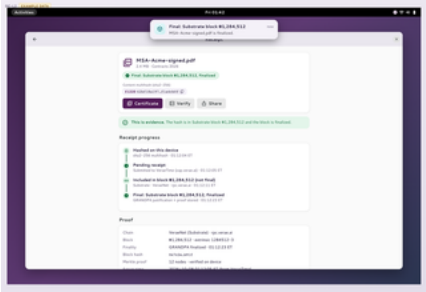
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-2 — DataCube list (home)

Home screen: all cubes with an at-a-glance evidence status per cube and a global 'receipts waiting' indicator.

ios	android	macos
		
DC-05	DC-A2	DC-M2

windows	debian	web
		
DC-W2	DC-L2	DC-B2

Features	
• Global pending banner (U-i3): count of not-final receipts with a jump to the Receipts queue.	• Filter chips incl. 'Attention'.
• Cube rows: cube glyph, size, chain health, status badge with text (never colour only).	• Sync + storage line (billing context).
• Tonal warning card for pending receipts.	• M3 filter chips with check icon when selected.
• Extended FAB 'New DataCube' (collapses on scroll); navigation bar with 5 destinations; predictive back (U-a3).	• Full-window drop target naming the destination (desktop_drop today).
• Progress HUD: per-file hashing (Rust core streaming), 'Already registered' instant receipt, pending state; cancel with ■.	• Toast (AppNotification) with Open receipt / Certificate.
• Tray flyout from the notification-area icon (gold dot = waiting).	• (1–5 as DC-08) in Cantarell / Adwaita styling. Notification via org.freedesktop.portal.Notification (works in Flatpak) — U-l3.

Function / source references
`lib/datacube/datacube_ui.dart CubeListScreen, DataCubesEdgeTab (empty text 'No DataCubes yet.')`
`lib/datacube/datacube_ui.dart CubeListScreen`
`file_browser.dart bottom bar`
`lib/files/file_browser.dart (desktop_drop)`
`lib/datacube/datacube_ui.dart (drag-and-drop, commit 79fd3bf)`
`New (U-w3)`
`lib/timestamps/timestamps_ui.dart`
`New`

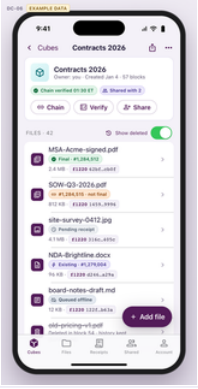
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-3 — Cube detail — files with receipt states

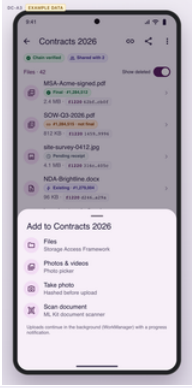
Shows a cube's files, each with its receipt state per George's rule, plus tombstoned (deleted) entries kept in history.

ios



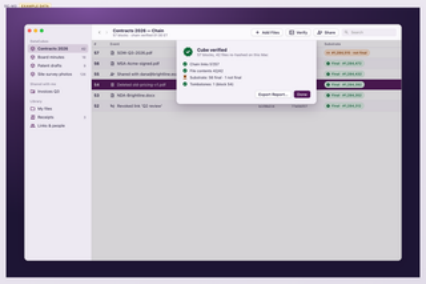
DC-06

android



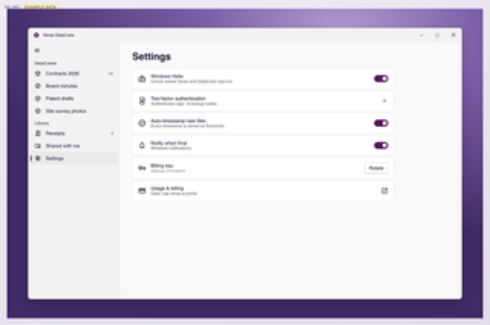
DC-A3

macos



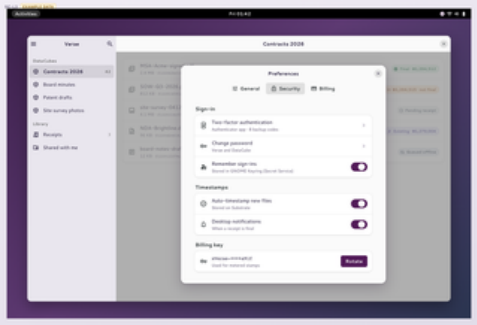
DC-M3

windows



DC-W3

debian



DC-L3

Features	
• Cube header: owner, block count, chain verification time, share count.	• Primary cube actions: Chain (DC-09), Verify (DC-10), Share (DC-11).
• 'Show deleted' toggle (CheckedPopupMenuItem 'Show deleted' today).	• File rows: name, size, short multihash, receipt badge: Final · #N / #N · not final / Pending receipt / Existing / Queued offline. Tombstones struck through with 'Deleted in block 54'.
• Add file → DC-07 (Files, Photos, Camera, Scan).	• Same receipt badges as iOS (shared component).
• M3 modal bottom sheet replaces the iOS action sheet.	• WorkManager for queued uploads and receipt reconcile (U-a5) with a foreground progress notification.
• Chain table with event type, prev/this hash, per-block Substrate state; tombstone row selectable.	• Window-modal sheet (HIG) with check list and Export Report.
• Settings cards; Windows Hello replaces Face ID (credentials in Credential Locker).	• AdwPreferencesDialog with pages General / Security / Billing.

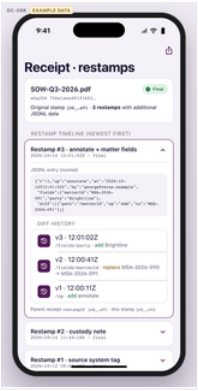
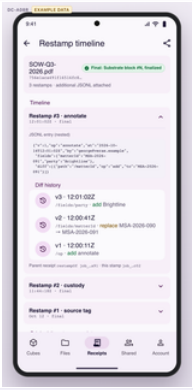
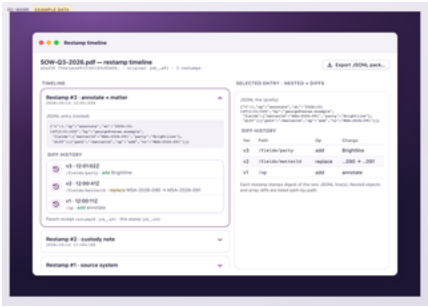
Function / source references
`lib/datacube/datacube_ui.dart CubeScreen (tombstoned rows, 'Show deleted', 'Timestamp un-stamped files')`
`lib/datacube/datacube_ui.dart CubeScreen`
`files/sync_queue.dart`
`lib/datacube/datacube_ui.dart CubeChainScreen, CubeVerifyScreen`
`lib/settings_screen.dart`
`lib/files/file_browser.dart`
`lib/timestamps/auto_stamp.dart`

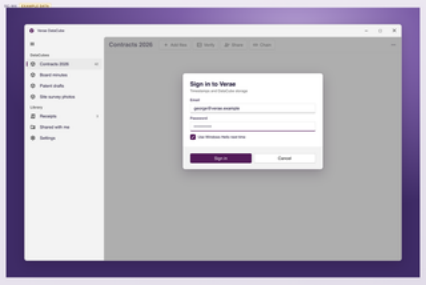
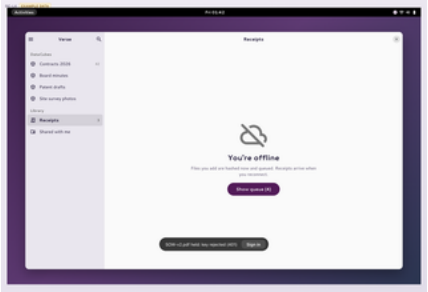
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`portal.2fa`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/2fa/*`	setup:POST {}, code:string, totpCode:string
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-4 — Receipt restamp timeline — expandable JSONL + diffs

Shows a file's original timestamp receipt and each restamp with additional JSONL payloads. Entries expand to nested JSON and per-field diff history.

ios	android	macos
		
DC-08R	DC-A08R	DC-M08R

windows	debian	web
		
DC-W4	DC-L4	DC-B2

Features	
• Header: file, sha256, restamp count.	• Timeline newest-first; expand/collapse.
• Expanded: pretty JSONL nested object + diff history list (path, op, from/to).	• restampOf link to parent job.
• Original receipt footer (no additional data).	• Contract note: restamp:true; digests of additional JSONL.
• M3 top app bar.	• Expandable cards.
• JSONL + diff list.	• Original footer.
• Split: timeline | detail.	• JSONL pretty + diff table.

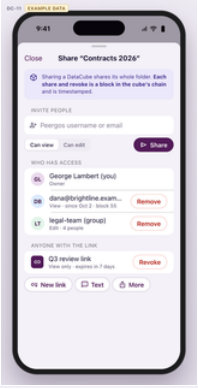
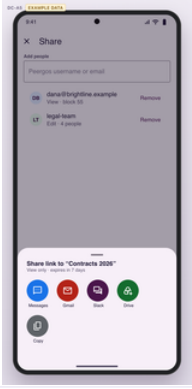
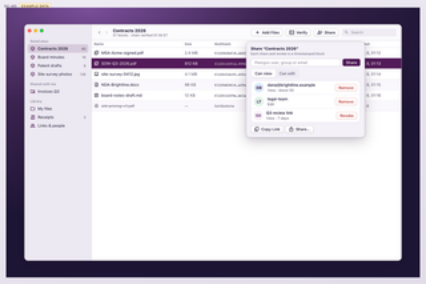
Function / source references
`lib/timestamps/*`
`CORE-CONTRACT restampOf`
`archive JSONL additional`
`lib/main.dart`
`lib/errors.dart`
`New`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-5 — Share & revoke (Peergos)

Share a cube or file with Peergos users/groups or a link; revoke access. Shares and revokes are cube-chain blocks with their own timestamps.

ios	android	macos
		
DC-11	DC-A5	DC-M5

Features	
• Chain-recording explanation (existing copy, elevated).	• Invite by Peergos username/email with view/edit segmented control.
• Access list with Remove (destructive style + confirmation).	• Links with expiry and Revoke.
• Channels: new link, SMS (existing 'Send by text'), iOS share sheet.	• Full-screen dialog pattern on compact width (M3).
• Android sharesheet with direct-share targets; revoke uses an M3 dialog (same copy as iOS).	• Popover anchored to the Share toolbar button; same content as iOS.
• System share menu (Mail, Messages, AirDrop).	

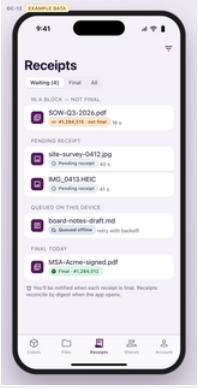
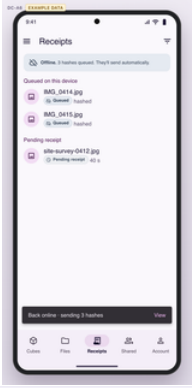
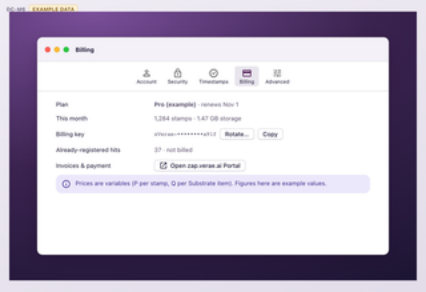
Function / source references
`lib/sharing/sharing_ui.dart ShareScreen, _NewLinkDialog, PeopleScreen, LinksScreen`
`lib/src/rust/api/sharing.dart revokeAccess`
`lib/sharing/sharing_ui.dart (share sheet)`
`lib/sharing/sharing_ui.dart`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-6 — Receipts queue (waiting for final)

A visible queue of every hash not yet final, grouped by stage, fixing U-i3 (pending jobs hidden until 'Verify' tapped).

ios	android	macos
		
DC-12	DC-A6	DC-M6

Features	
• Segment: Waiting / Final / All.	• Groups by stage — included (not final), pending, queued offline, final today.
• Offline queue items with retry/backoff from auto_stamp.dart's persistent queue.	• Push (EP-044) + reconcile-on-launch explanation.
• Offline tonal card; snackbar on reconnect with action.	• Settings toolbar tabs: Account, Security, Timestamps, Billing, Advanced.
• Billing pane: plan, usage, billing key rotate/copy, portal link.	• Price-variable note.

Function / source references
`lib/timestamps/timestamps_ui.dart TimestampsScreen`
`lib/timestamps/auto_stamp.dart (persistent queue, backoff)`
`lib/files/sync_queue.dart`
`lib/timestamps/auto_stamp.dart`
`lib/errors.dart`
`lib/settings_screen.dart`
`lib/billing/usage_billing_screen.dart`

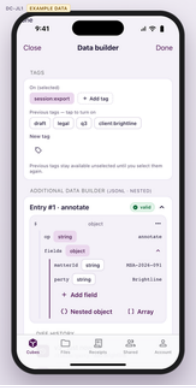
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-JL1 — Expandable JSONL data builder + previous tags

Full-screen nested JSONL builder; previous tags selectable; diff history.

ios



DC-JL1

Features	
• Previous tags.	• Nested builder.
• Diffs.	• Use in flow.

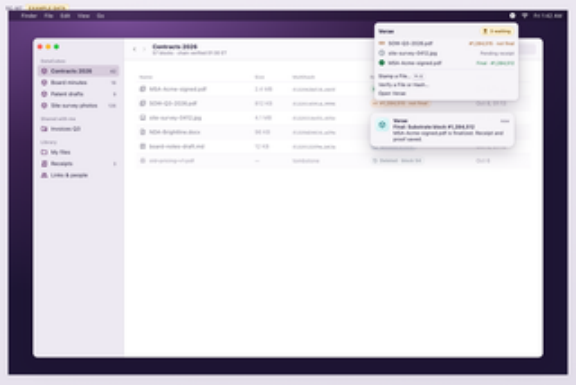
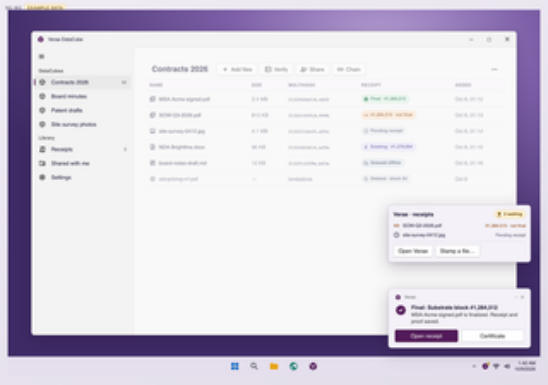
Function / source references
`lib/files/*`
`lib/datacube/*`
`lib/timestamps/*`
`peergos`
`customer.verae.ai/v1/timestamp`
`lib/timestamps/* (restamp JSONL)`
`design DC-JL1 builder`

API id	Method	Path	Key params (name:type)
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-7 — Menu bar extra + notifications

Ambient status on macOS: a menu bar extra with waiting receipts and quick actions; User Notifications when a receipt is final.

macos	windows
	
DC-M7	DC-W2

Features	
• Menu bar extra (MenuBarExtra) listing not-final receipts with stage text; global hotkey to stamp.	• Notification on finality with Open / Certificate actions.
• Toast (AppNotification) with Open receipt / Certificate.	• Tray flyout from the notification-area icon (gold dot = waiting).

Function / source references
`New (Mac equivalent of U-w3/U-l3)`
`New (U-w3)`

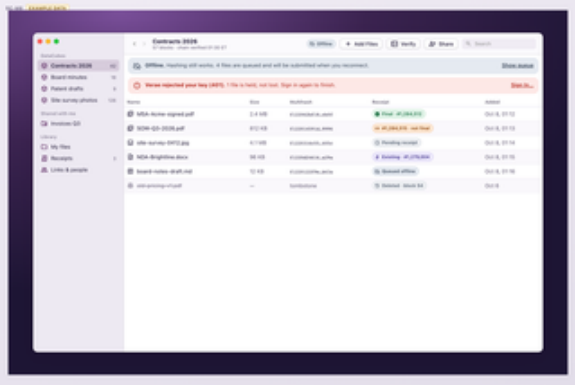
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

DC-8 — Offline & error states (Mac)

DC-15 on Mac: stacked banners above the table; toolbar offline pill.

macos



DC-M8

Features	
• Offline banner with queue link.	• Action-needed banner; held items.

Function / source references
`lib/errors.dart`
`lib/files/sync_queue.dart`

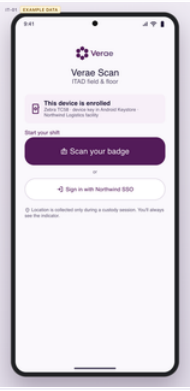
API id	Method	Path	Key params (name:type)
`peergos.cube`	—	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-01 — Device enrollment & shift sign-in

Per-shift sign-in on an enrolled device. Events are signed by the device key and carry actor + device (doc 02 §1).

android



IT-01

Features	
• Enrolled-device card (authentik OIDC device flow; Ed25519 key in Keystore / Secure Enclave).	• Badge scan sign-in: 72 dp primary target for gloves.
• SSO alternative.	• Location privacy statement (doc 02 'Privacy').

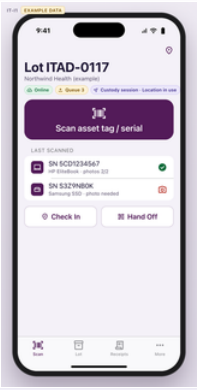
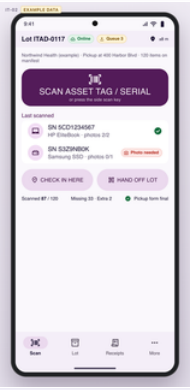
Function / source references
`itad-provenance/docs/02 §1 (Who)`
`10`
`peergos-flutter lib/auth/*`

API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-02 — Field home — iOS

IT-02 with HIG: large title, tab bar, Live Activity for the custody session, and the blue location indicator.

ios	android
	
IT-I1	IT-02

Features	
• Status badges; iOS shows the system location indicator + a Live Activity on the Lock Screen during custody sessions.	• Large scan button (VisionKit DataScannerViewController).
• Secondary actions as bordered buttons; no hardware scan key on iPhone (MFi ring scanners optional).	• Lot context: customer, site, manifest count.
• 96 dp scan button + hardware scan key (Zebra DataWedge / Honeywell intents).	• Last scanned list with evidence completeness (photos needed).
• Check-in (GPS GeoFix + server time) and Hand off (QR) as large tonal actions.	• Live counts and the pickup form's receipt status.

Function / source references
`itad-provenance/docs/02 §1`
`22 §2.1`
`itad-provenance/docs/22 §4.1`
`02 §1`
`13 §3–4`

API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-03 — Scanner — iOS (VisionKit)

IT-03 on iOS: DataScannerViewController with item highlighting; result in a sheet.

ios	android
IT-I2	IT-03

Features	
• VisionKit highlights recognised codes/text; tap to choose when several.	• Result sheet with HIG buttons; haptic (UINotificationFeedbackGenerator).
• Viewfinder with scan line; works with gloves (no tap needed).	• Supported symbologies; NFC toggle for NTAG labels.
• Result sheet: matched manifest line, or 'Not on manifest → Extra'.	• Large next actions; photo hashing on capture.

Function / source references
`itad-provenance/docs/02 §1`
`22 §2.1`
`itad-provenance/docs/02 §1 (Identify)`
`22 U-i7/U-a2 (no scanner today)`

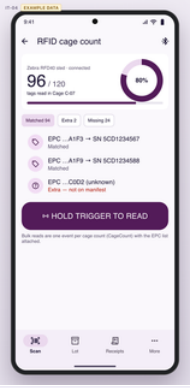
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-04 — RFID bulk read (cage count)

BLE UHF RFID sled reads a whole cage; results reconcile against the manifest.

android



IT-04

Features	
• Live count vs manifest with progress ring (number + text).	• Filter chips: matched / extra / missing.
• Trigger-to-read action (hardware trigger on sled).	

Function / source references
`itad-provenance/docs/02 §1 (BLE UHF RFID sleds)`

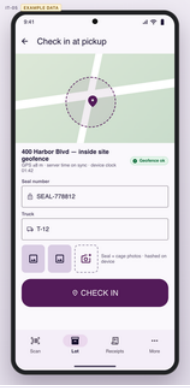
API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-05 — Pickup check-in (GPS GeoFix)

Records arrival at the pickup site with GPS accuracy, seal, truck and photos; starts the custody session check-ins (every 5 min by default).

android



IT-05

Features	
• Map with geofence and accuracy circle.	• Geofence status + time source note (server time on sync; device time is captured_at only).
• Seal and truck fields (scan-to-fill).	• Photo evidence hashed on device (SHA-256 in the event).

Function / source references
`itad-provenance/docs/02 §1 (Where, custody check-in sequence)`
`13 §4`

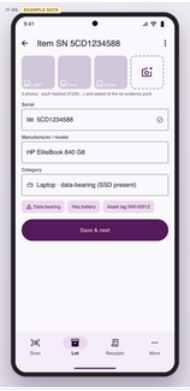
API id	Method	Path	Key params (name:type)
`peergos.cube`	—	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-06 — Item capture — photos & identity

Per-item evidence: photos (hashed on device), serial, model and flags; feeds Intake (doc 13 §6).

android



IT-06

Features	
• Photo strip with required slots (label, front, screen).	• Identity fields; serial confirmed from scan.
• Flags: data-bearing, battery (hazmat), asset tag alias.	

Function / source references
`itad-provenance/docs/02 §1 (Evidence)`
`13 §6`
`lib/files/file_browser.dart`
`lib/timestamps/auto_stamp.dart`

API id	Method	Path	Key params (name:type)
`peergos.cube`	—	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-07 — Custody hand-off — QR exchange

Two-party custody transfer (driver ↔ dock). The giver shows a one-time QR; the receiver scans; both sign (gap 5 in doc 22 §3).

android

IT-07

Features	
• One-time QR with lot, transfer ULID and nonce.	• Summary of what is being transferred (items, seal state, parties).
• Explains signing and receipt.	• NFC tap alternative.

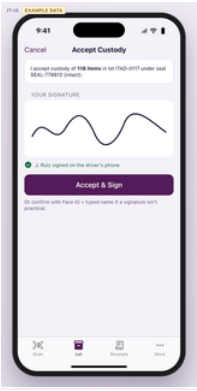
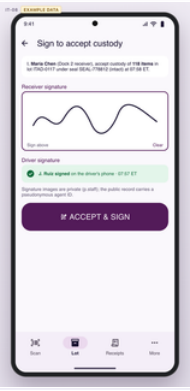
Function / source references
`itad-provenance/docs/13 §4`
`02 §1 (custody check-in sequence)`
`22 §3 gap 5`
`lib/files/file_browser.dart`
`lib/timestamps/auto_stamp.dart`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-08 — Signature — iOS (PencilKit)

IT-08 on iOS using PencilKit canvas; Apple Pencil on iPad dock tablets.

ios	android
	
IT-I3	IT-08

Features	
• PencilKit canvas; VoiceOver alternative: typed name + Face ID confirmation.	• Primary action at bottom per HIG.
• Plain-language statement of what is accepted.	• Signature pad with clear; large target.
• Other party's status (signed on their device).	• Privacy note (doc 07 §4.1 p.staff).

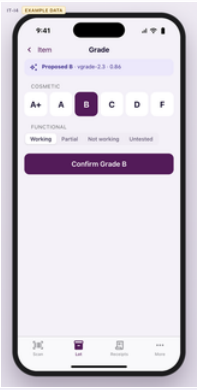
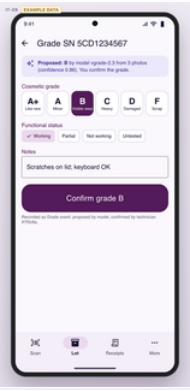
Function / source references
`itad-provenance/docs/02 §1`
`22 §2.1`
`itad-provenance/docs/07 §4.1`
`13 §4, §11, §17 (signature on phone)`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-09 — Grading — iOS

IT-09 on iOS: segmented control for functional status; same grade scale.

ios	android
	
IT-I4	**IT-09**

Features	
• Grade buttons ≥ 58 pt.	• UISegmentedControl for functional status.
• AI proposal banner with model/version and confidence (DecisionAgent input).	• Six large grade targets A+…F with descriptions (not colour only).
• Functional status chips.	• Confirm action names the grade.

Function / source references
`itad-provenance/docs/02 §1`
`22 §2.1`
`itad-provenance/docs/02 §2.2`
`13 §6`

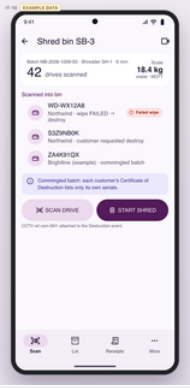
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-10 — Floor mode — shred bin (and wipe check)

Floor use of the mobile app: scan serials into a shred bin with scale reading and CCTV reference; failed wipes flow here (doc 13 §7–8). The full wipe/shred stations are IT-S5/S6.

android



IT-10

Features	
• Batch header with machine, particle size, scale reading (MQTT, stable).	• Serial list with reason (failed wipe / requested destroy).
• Commingled-batch rule explained.	• Scan / Start shred actions.

Function / source references
`itad-provenance/docs/13 §7–8`
`02 §2 (shredder station)`

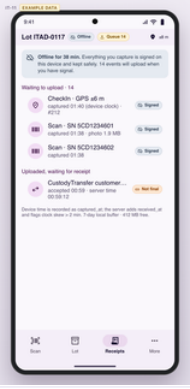
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-11 — Offline queue

Offline-first capture: signed events queue locally (SQLite) and upload to itad-mobile-sync; status shows what is safe vs still waiting.

android



IT-11

Features	
• Persistent offline banner with duration and reassurance.	• Queue of signed events (device key) with capture time and size.
• Time semantics (captured_at vs received_at) and buffer capacity.	

Function / source references
`itad-provenance/docs/02 §1 (Offline)`
`22 §3 gap 8`

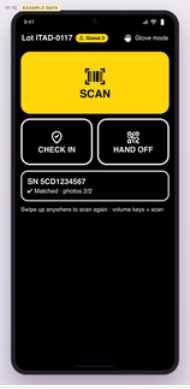
API id	Method	Path	Key params (name:type)
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-12 — Glove / outdoor (sunlight) mode

High-contrast, oversized layout for gloves and direct sunlight: ≥ 120 dp targets, black/yellow palette (21:1 / 19.6:1 contrast), haptics, hardware keys.

android



IT-12

Features	
• Giant scan target; one-hand reach.	• Two large secondary actions.
• Last scan readout in 20 sp+ type with symbol + word.	• Gesture and hardware-key alternatives.

Function / source references

`itad-provenance/docs/22 U-i9, U-a6 (glove-size targets, sunlight theme)`

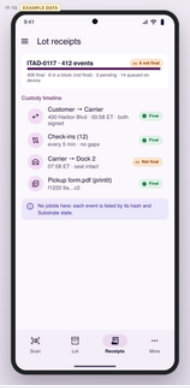
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-13 — Lot receipts & custody timeline

Per-lot evidence status: every event's receipt state, custody legs and documents, using the shared receipt badge.

android



IT-13

Features	
• Lot evidence progress (final vs not).	• Custody timeline rows with state.
• Rule reminder.	

Function / source references
`itad-provenance/docs/06`
`16 §6`
`22 §4.4`

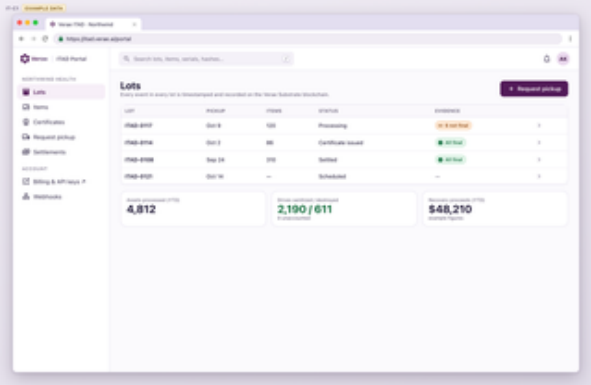
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-C1 — Customer portal — lots

Customer view of lots with lifecycle status and evidence state; entry to items, certificates and settlements (doc 21 §10).

web



IT-C1

Features	
• Lots table with evidence column (receipt states roll-up).	• Summary KPIs.

Function / source references
`itad-provenance/docs/21 §10`
`15 §7`

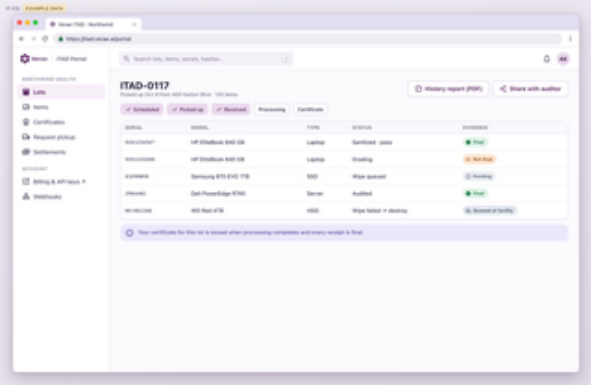
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-C2 — Customer portal — lot detail

Lot progress and items with per-item status and evidence state.

web



IT-C2

Features	
• Lifecycle chips.	• Items table with status and evidence.
• Certificate timing explanation.	

Function / source references
`itad-provenance/docs/21 §10`

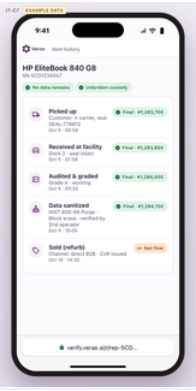
API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-C3 — Item history — mobile web

IT-C3 reflowed for phones (buyers scanning an asset's QR label).

web



IT-C7

Features	
• Summary badges wrap.	• Single-column history.

Function / source references

`itad-provenance/docs/09 §4`

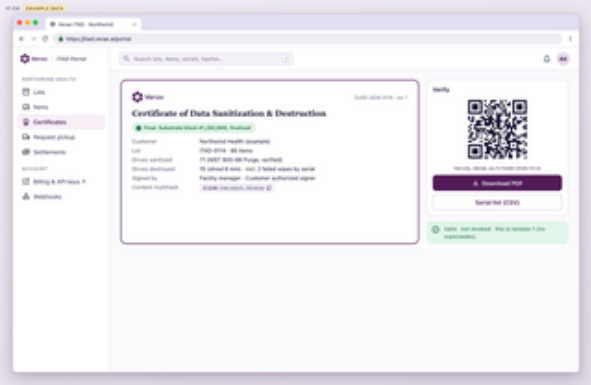
API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-C4 — Certificate view + verify

Customer certificate with final receipt, signers, verify QR, downloads and validity/revocation status (doc 15 §4–5).

web



IT-C4

Features	
• Certificate preview with receipt state, signers and multihash.	• Verify QR + downloads.
• Validity / revocation / revision status.	

Function / source references
`itad-provenance/docs/15 §4–5`
`13 §23`

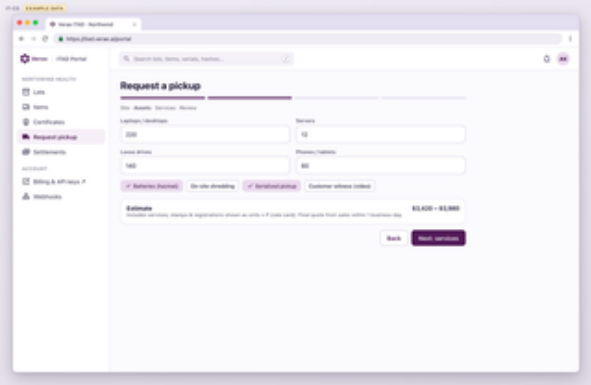
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-C5 — Request pickup & quote

Pickup request wizard with an estimate; becomes a sales quote (EP-110).

web



IT-C5

Features	
• Stepper.	• Asset counts.
• Service options incl. hazmat and witness.	• Estimate with units × P disclaimer.

Function / source references
`itad-provenance/docs/21 §5, §10`
`13 §1`

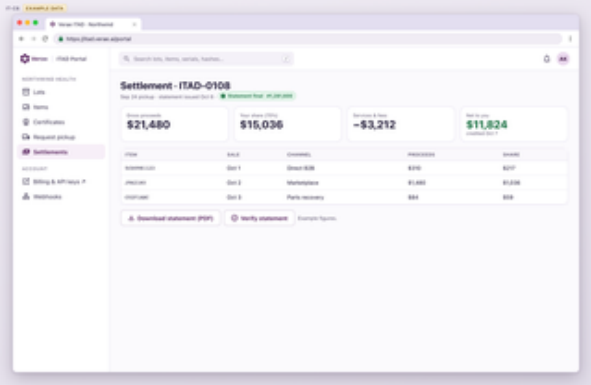
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-C6 — Settlement statements

Asset-recovery settlement: proceeds × share – fees, each line tied to a sale event; the statement itself is stamped and verifiable (doc 21 §3).

web



IT-C6

Features	
• Summary math.	• Per-item lines referencing sale events.
• Download + verify.	

Function / source references
`itad-provenance/docs/21 §3`

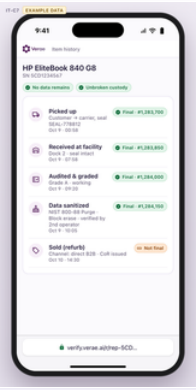
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-C7 — Item history — mobile web

IT-C3 reflowed for phones (buyers scanning an asset's QR label).

web



IT-C7

Features	
• Summary badges wrap.	• Single-column history.

Function / source references

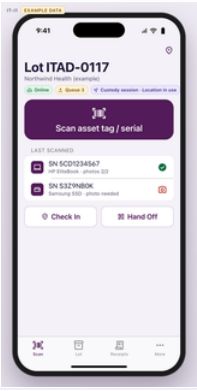
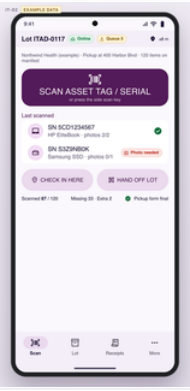
`itad-provenance/docs/09 §4`

API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-I1 — Field home — iOS

IT-02 with HIG: large title, tab bar, Live Activity for the custody session, and the blue location indicator.

ios	android
	
IT-I1	IT-02

Features	
• Status badges; iOS shows the system location indicator + a Live Activity on the Lock Screen during custody sessions.	• Large scan button (VisionKit DataScannerViewController).
• Secondary actions as bordered buttons; no hardware scan key on iPhone (MFi ring scanners optional).	• Lot context: customer, site, manifest count.
• 96 dp scan button + hardware scan key (Zebra DataWedge / Honeywell intents).	• Last scanned list with evidence completeness (photos needed).
• Check-in (GPS GeoFix + server time) and Hand off (QR) as large tonal actions.	• Live counts and the pickup form's receipt status.

Function / source references
`itad-provenance/docs/02 §1`
`22 §2.1`
`itad-provenance/docs/22 §4.1`
`02 §1`
`13 §3–4`

API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-I2 — Scanner — iOS (VisionKit)

IT-03 on iOS: DataScannerViewController with item highlighting; result in a sheet.

ios	android
IT-I2	IT-03

Features	
• VisionKit highlights recognised codes/text; tap to choose when several.	• Result sheet with HIG buttons; haptic (UINotificationFeedbackGenerator).
• Viewfinder with scan line; works with gloves (no tap needed).	• Supported symbologies; NFC toggle for NTAG labels.
• Result sheet: matched manifest line, or 'Not on manifest → Extra'.	• Large next actions; photo hashing on capture.

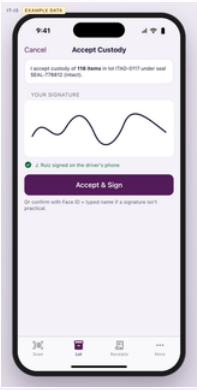
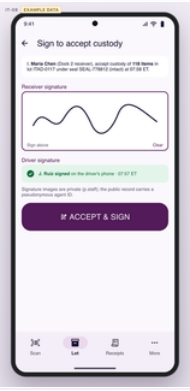
Function / source references
`itad-provenance/docs/02 §1`
`22 §2.1`
`itad-provenance/docs/02 §1 (Identify)`
`22 U-i7/U-a2 (no scanner today)`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-I3 — Signature — iOS (PencilKit)

IT-08 on iOS using PencilKit canvas; Apple Pencil on iPad dock tablets.

ios	android
	
IT-I3	IT-08

Features	
• PencilKit canvas; VoiceOver alternative: typed name + Face ID confirmation.	• Primary action at bottom per HIG.
• Plain-language statement of what is accepted.	• Signature pad with clear; large target.
• Other party's status (signed on their device).	• Privacy note (doc 07 §4.1 p.staff).

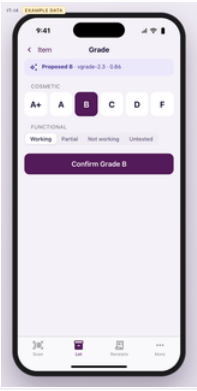
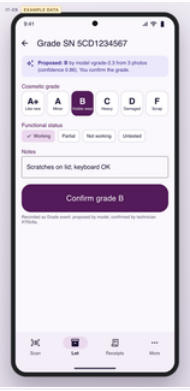
Function / source references
`itad-provenance/docs/02 §1`
`22 §2.1`
`itad-provenance/docs/07 §4.1`
`13 §4, §11, §17 (signature on phone)`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-I4 — Grading — iOS

IT-09 on iOS: segmented control for functional status; same grade scale.

ios	android
	
IT-I4	IT-09

Features	
• Grade buttons ≥ 58 pt.	• UISegmentedControl for functional status.
• AI proposal banner with model/version and confidence (DecisionAgent input).	• Six large grade targets A+…F with descriptions (not colour only).
• Functional status chips.	• Confirm action names the grade.

Function / source references
`itad-provenance/docs/02 §1`
`22 §2.1`
`itad-provenance/docs/02 §2.2`
`13 §6`

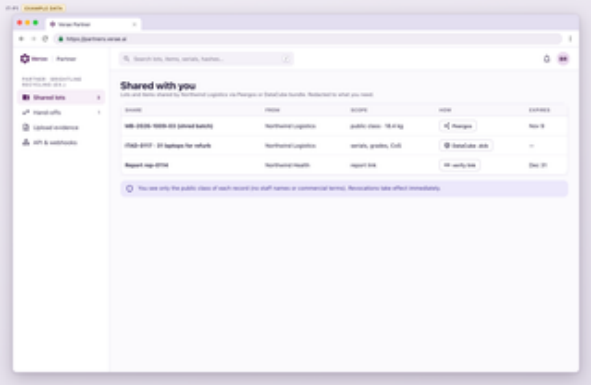
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-P1 — Partner — shared lots

Downstream processors/buyers see what was shared (Peergos share, DataCube bundle, or report link), redacted (doc 07, 21 §9).

web



IT-P1

Features	
• Share list with channel and expiry.	• Redaction/revocation explanation.

Function / source references
`itad-provenance/docs/07 §2–6`
`21 §9`

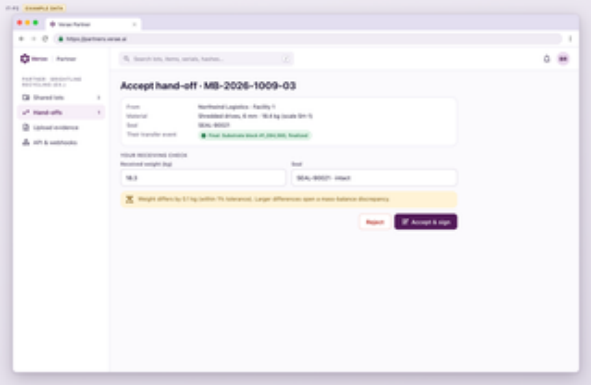
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-P2 — Partner — accept custody hand-off

A partner accepts a custody transfer as a signed event (doc 21 §9), with mass-balance check.

web



IT-P2

Features	
• Transfer summary with sender's receipt state.	• Receiving check fields.
• Tolerance rule (doc 13 §13 mass balance).	

Function / source references
`itad-provenance/docs/21 §9`
`13 §13, §20`

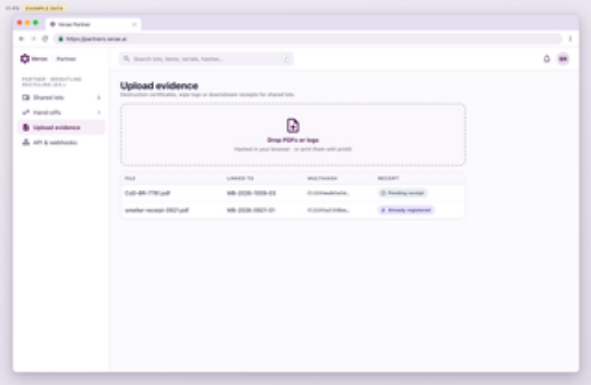
API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-P3 — Partner — upload evidence

Partners add their own evidence (EP-109), hashed client-side, stamped and linked to shared lots.

web



IT-P3

Features	
• Drop zone with client-side hashing.	• Upload table with receipt states incl. already-registered.

Function / source references

`itad-provenance/docs/21 §9 (EP-109)`

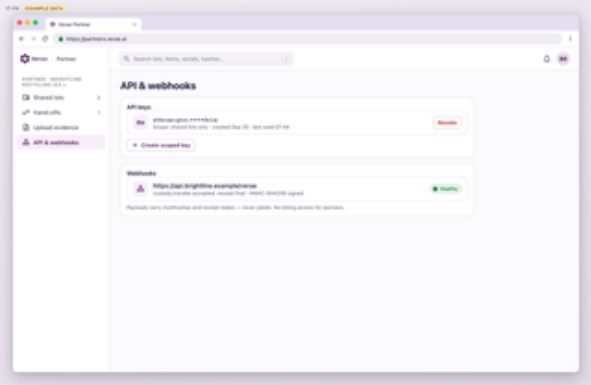
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-P4 — Partner — API keys & webhooks

Partner keys scoped to shared lots (authz EP-060) and HMAC-signed webhooks (EP-108).

web



IT-P4

Features	
• Scoped keys with revoke.	• Webhooks with event list and health.

Function / source references
`itad-provenance/docs/21 §9`
`25 (EP-060, EP-108)`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-S1 — Station kiosk sign-in (badge + PIN)

Dock-PC kiosk sign-in (U-w6): station has its own device key; operators badge in per shift.

windows

IT-S1

Features	
• Station identity + leaf-node health.	• NFC/HID badge reader; barcode badge fallback.
• PIN (second factor).	• Kiosk policy statement.

Function / source references
`itad-provenance/docs/02 §2 (stations thin, device key)`
`22 U-w6`

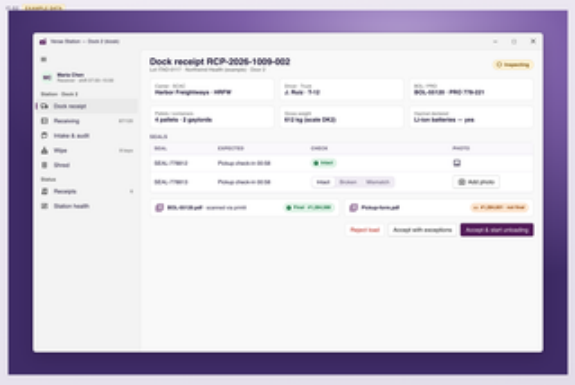
API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-S2 — Dock receipt (shipment level)

Records the truck before any item scan: carrier, BOL/PRO, seals, pallets, weight, hazmat; accept / with exceptions / reject (doc 13 §5).

windows



IT-S2

Features	
• Projected receipt status (scheduled → arrived → inspecting → accepted…).	• Shipment facts grid (scan-to-fill where possible).
• Seal checks with photos; mismatch → Discrepancy.	• Documents with receipt states (BOL printed through printit).
• Decision actions; reject sends custody back to carrier.	

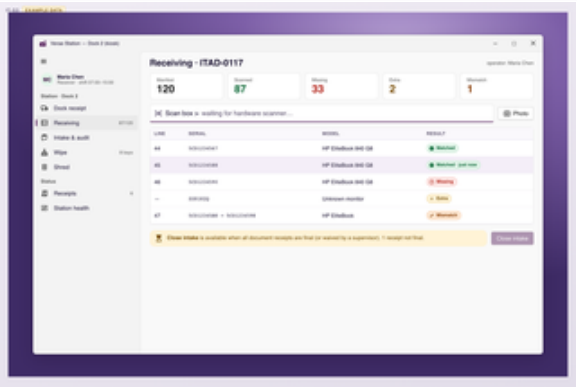
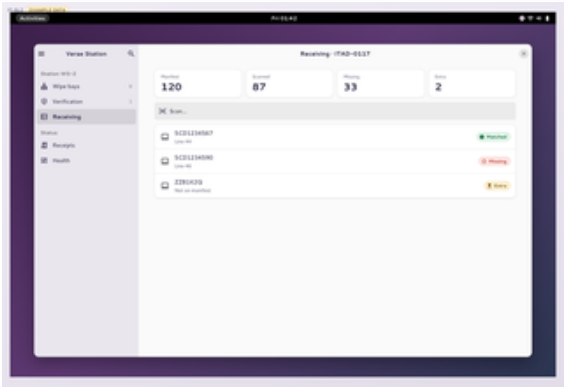
Function / source references
`itad-provenance/docs/13 §5`
`02 §2 (dock receipt)`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-S3 — Receiving reconciliation vs manifest

Item scans matched to the manifest/ASN with missing / extra / mismatch discrepancies (doc 13 §5, doc 22 §4.2).

windows	debian
	
IT-S3	**IT-SL2**

Features	
• Counters with numbers + labels (not colour only).	• Hardware-scanner input box keeps focus; audible + visual confirmation.
• Result table; each discrepancy becomes a Discrepancy event with resolution.	• Close intake gated on final receipts.
• Counters.	• Scanner entry keeps focus.

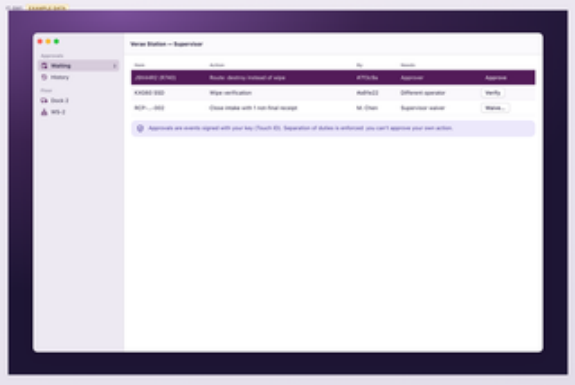
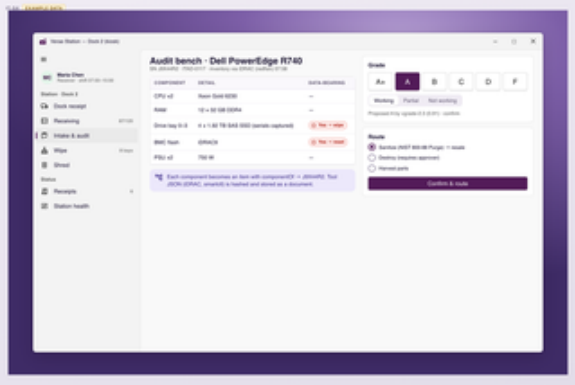
Function / source references
`itad-provenance/docs/13 §5`
`22 §4.2`

API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-S4 — Supervisor approvals (Mac)

Station-family app on a supervisor's Mac: approve route changes, co-sign verifications, waive close-intake gates.

macos	windows
	
IT-SM1	IT-S4

Features	
• Approval queue (Approver / Verifier / Supervisor roles).	• Touch ID signing + separation-of-duties rule.
• Component inventory parsed from iDRAC/iLO/lshw/smartctl JSON.	• componentOf explanation; tool output hashed.
• Grade with AI proposal confirmation.	• Route; destroy routes need a human approver (vitad:Approver).

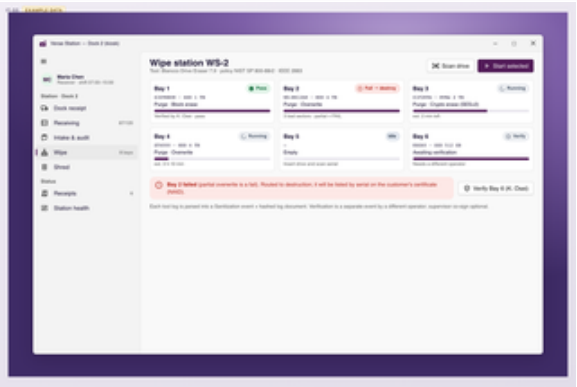
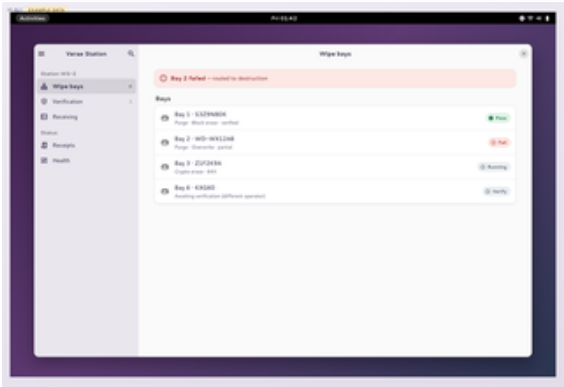
Function / source references
`itad-provenance/docs/02 §2 (routing approver, supervisor co-sign)`
`13 §24`
`itad-provenance/docs/13 §6, §24`
`02 §2 (audit/test bench)`

API id	Method	Path	Key params (name:type)
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-S5 — Wipe station (per-drive bays)

Drive-bay dashboard: method per NIST/IEEE, progress, pass/fail, separate-operator verification, failed drives routed to destroy (doc 13 §7).

windows	debian
	
IT-S5	**IT-SL1**

Features	
• Bay cards with serial, method/technique, progress and result badge.	• Failure handling rule (partial = fail; NAID serial listing).
• Verification by a different operator (enforced).	• Log → event explanation.
• AdwBanner (error) for failures.	• Boxed list rows per bay; keyboard: arrows + Enter to open bay detail.

Function / source references
`itad-provenance/docs/13 §7`
`02 §2 (wipe stations)`

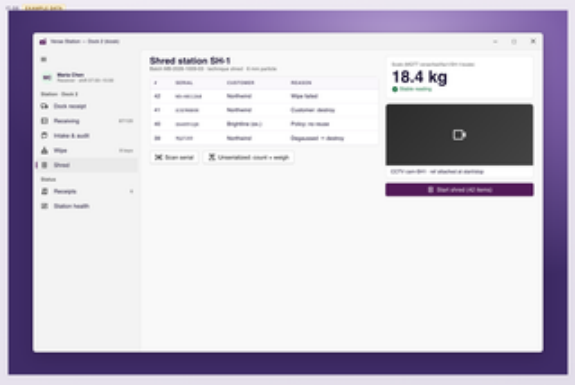
API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-S6 — Shred / destruction station

Serial-scanned destruction with scale, particle size and CCTV reference; produces the material batch and per-customer Certificates of Destruction (doc 13 §8).

windows



IT-S6

Features	
• Serial log back to each client (NAID).	• Stable scale reading via MQTT becomes weightKg on the confirmed event.
• CCTV reference.	• Start shred = Destruction event (inputs invalidated, batch generated).

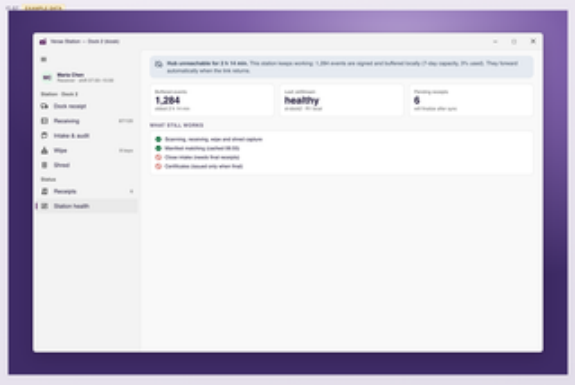
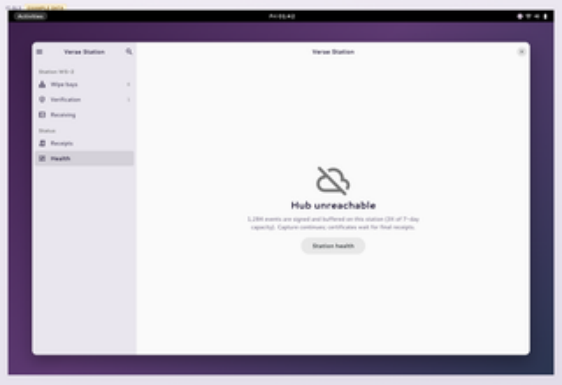
Function / source references
`itad-provenance/docs/13 §8`
`02 §2 (shredder station, MQTT)`

API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-S7 — Station offline (leaf buffering)

Station-level offline state: the embedded NATS leaf buffers 7 days; operators see what works and what waits for finality.

windows	debian
	
IT-S7	IT-SL3

Features	
• Banner with duration and buffer use.	• Health tiles.
• Works / waits list.	• AdwStatusPage with counts.

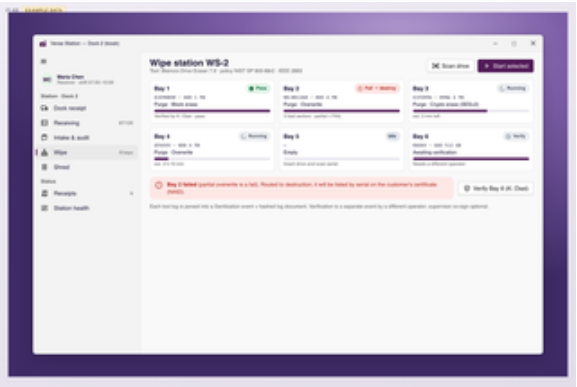
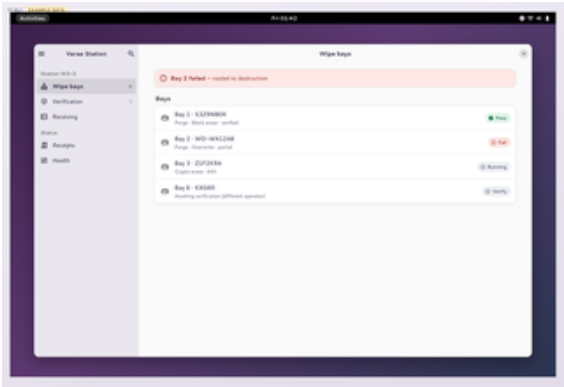
Function / source references
`itad-provenance/docs/02 §2 (stations thin, 7-day buffer)`
`03`
`11`
`itad-provenance/docs/02 §2`

API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-SL1 — Wipe station (per-drive bays)

Drive-bay dashboard: method per NIST/IEEE, progress, pass/fail, separate-operator verification, failed drives routed to destroy (doc 13 §7).

windows	debian
	
IT-S5	**IT-SL1**

Features	
• Bay cards with serial, method/technique, progress and result badge.	• Failure handling rule (partial = fail; NAID serial listing).
• Verification by a different operator (enforced).	• Log → event explanation.
• AdwBanner (error) for failures.	• Boxed list rows per bay; keyboard: arrows + Enter to open bay detail.

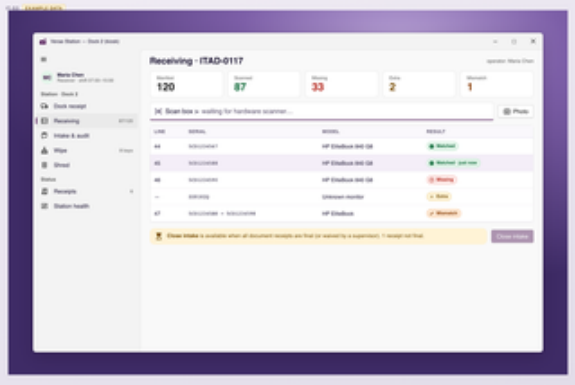
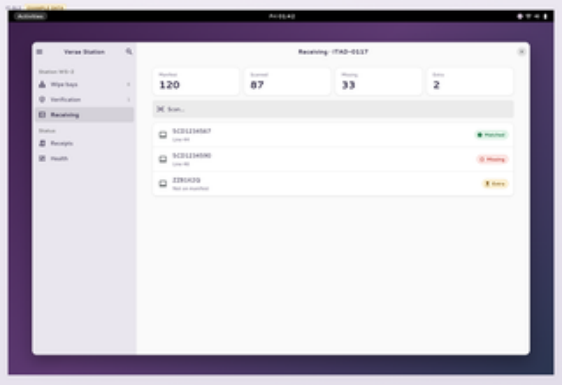
Function / source references
`itad-provenance/docs/13 §7`
`02 §2 (wipe stations)`

API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-SL2 — Receiving reconciliation vs manifest

Item scans matched to the manifest/ASN with missing / extra / mismatch discrepancies (doc 13 §5, doc 22 §4.2).

windows	debian
	
IT-S3	**IT-SL2**

Features	
• Counters with numbers + labels (not colour only).	• Hardware-scanner input box keeps focus; audible + visual confirmation.
• Result table; each discrepancy becomes a Discrepancy event with resolution.	• Close intake gated on final receipts.
• Counters.	• Scanner entry keeps focus.

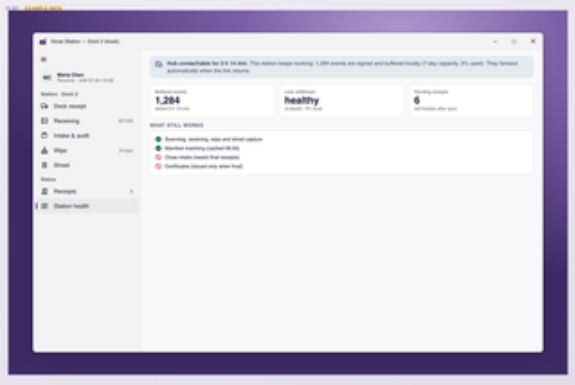
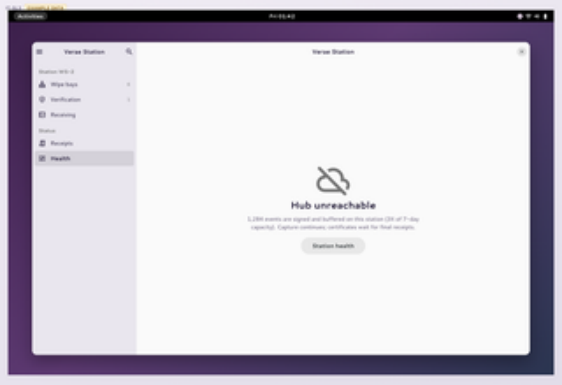
Function / source references
`itad-provenance/docs/13 §5`
`22 §4.2`

API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-SL3 — Station offline (leaf buffering)

Station-level offline state: the embedded NATS leaf buffers 7 days; operators see what works and what waits for finality.

windows	debian
	
IT-S7	IT-SL3

Features	
• Banner with duration and buffer use.	• Health tiles.
• Works / waits list.	• AdwStatusPage with counts.

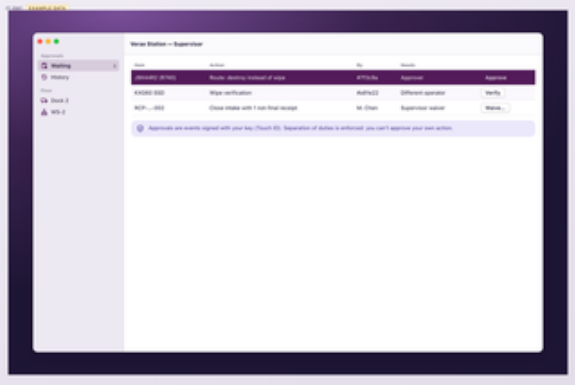
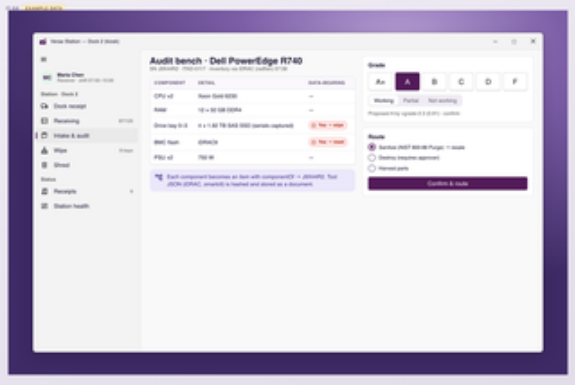
Function / source references
`itad-provenance/docs/02 §2 (stations thin, 7-day buffer)`
`03`
`11`
`itad-provenance/docs/02 §2`

API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-SM1 — Supervisor approvals (Mac)

Station-family app on a supervisor's Mac: approve route changes, co-sign verifications, waive close-intake gates.

macos	windows
	
IT-SM1	**IT-S4**

Features	
• Approval queue (Approver / Verifier / Supervisor roles).	• Touch ID signing + separation-of-duties rule.
• Component inventory parsed from iDRAC/iLO/lshw/smartctl JSON.	• componentOf explanation; tool output hashed.
• Grade with AI proposal confirmation.	• Route; destroy routes need a human approver (vitad:Approver).

Function / source references
`itad-provenance/docs/02 §2 (routing approver, supervisor co-sign)`
`13 §24`
`itad-provenance/docs/13 §6, §24`
`02 §2 (audit/test bench)`

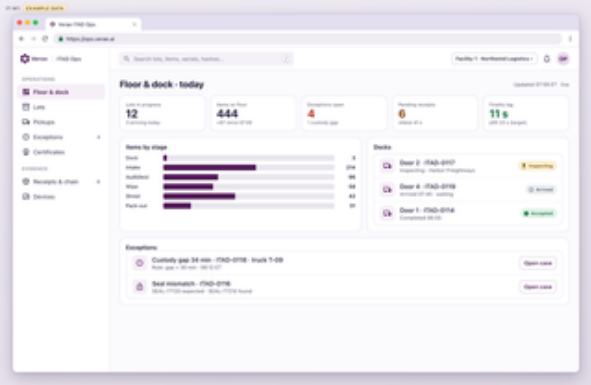
API id	Method	Path	Key params (name:type)
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-W1 — Ops console — floor & dock dashboard

Operations overview: lots in progress, items by stage, docks, exceptions, pending-receipt depth and finality lag (doc 21 §7).

web



IT-W1

Features	
• KPIs incl. pending receipts and finality lag vs the p95 22 s target (doc 24).	• Items by stage bars (with numbers).
• Dock door status.	• Exceptions list with case action.

Function / source references
`itad-provenance/docs/21 §7`
`24`
`13 §22`

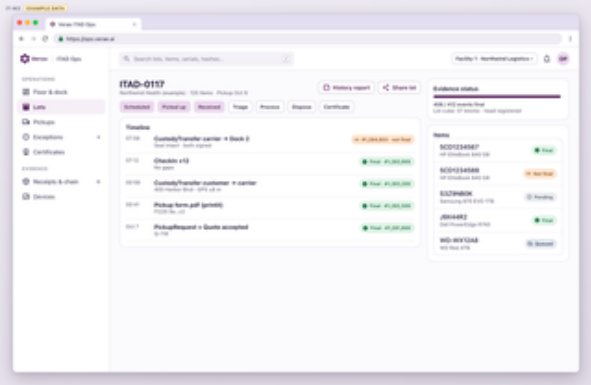
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-W2 — Lot detail — timeline & evidence

One lot: lifecycle stage, event timeline with per-event receipt states, evidence progress and items.

web



IT-W2

Features	
• Lifecycle stage chips (Intake → Triage → Process → Dispose → Certificate).	• Timeline with Substrate state per event.
• Evidence progress and lot-cube status.	• Items list → item history.

Function / source references
`itad-provenance/docs/13`
`06`
`17`

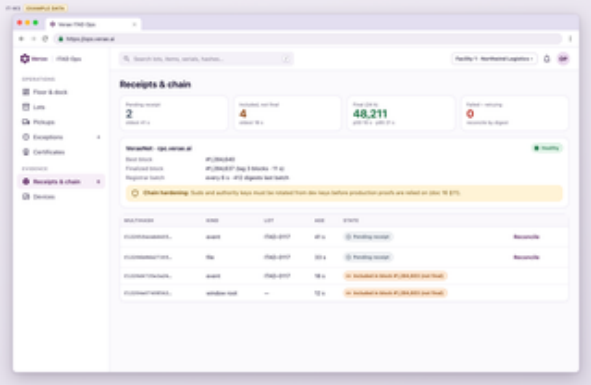
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-W3 — Pending-receipt queue & chain status

Queue depth and age by stage, chain health (finality lag), hardening warning, and per-digest reconcile (never blind resubmit).

web



IT-W3

Features	
• Stage counters with latency percentiles (doc 24 targets are estimates).	• Chain status (best vs finalized block) + hardening warning (doc 16 §11).
• Per-digest table; Reconcile runs lookup-by-digest.	

Function / source references
`itad-provenance/docs/24`
`16 §6, §11`
`21 §7`

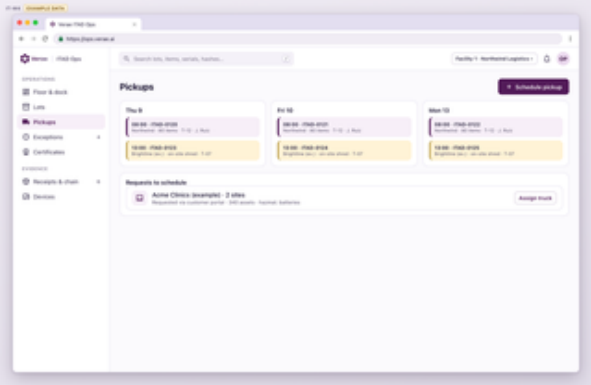
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-W4 — Pickups — schedule & assign trucks

Schedule pickups, assign trucks/drivers, and process requests from the customer portal (doc 21 §7, 13 §1).

web



IT-W4

Features	
• Calendar cards per day with truck + driver.	• Inbound requests (EP-110) with assign action.

Function / source references
`itad-provenance/docs/13 §1`
`21 §5, §7`

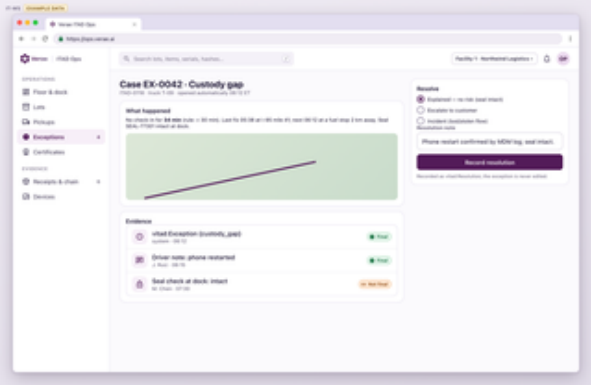
API id	Method	Path	Key params (name:type)
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-W5 — Exceptions & cases

Custody gaps, seal mismatches, discrepancies: evidence and an append-only resolution (corrections are new events).

web



IT-W5

Features	
• Narrative + map of the gap.	• Evidence events with states.
• Resolution options; recorded as a new event.	

Function / source references
`itad-provenance/docs/13 §4, §18`
`21 §4`

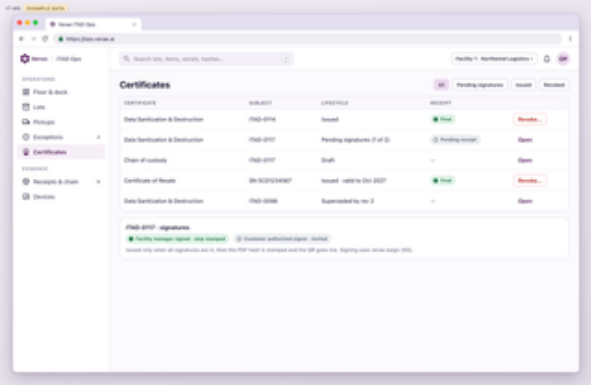
API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

IT-W6 — Certificate lifecycle

Draft → pending signatures → issued → revoked / superseded / expired; each signature and the issue are stamped (doc 13 §23, 15 §5).

web



IT-W6

Features	
• Certificate register with lifecycle and receipt state; revoke needs cert.revoke.	• Signature progress (via the signing round-trip).

Function / source references
`itad-provenance/docs/13 §23`
`15 §5`

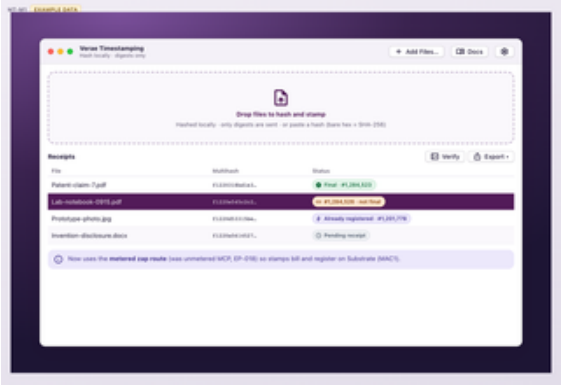
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

MT-M1 — Verae Timestamping (Mac) — drop, hash, receipts

Redesign of the existing SwiftUI app (ContentView.swift: drop files, Register/Verify, receipt list, export). Adds Substrate receipt states and the metered route.

macos



MT-M1

Features	
• Drop zone + paste-a-hash; bare digest treated as SHA-256.	• Receipt table with the 4-state model (replaces register/verify result text).
• Route change note (MCP → metered zap).	

Function / source references

`verae-mcp-timestamping-client/mac/VeraeTimestamping/ContentView.swift, MCPClient.swift, FileHasher.swift, ReceiptStore.swift`

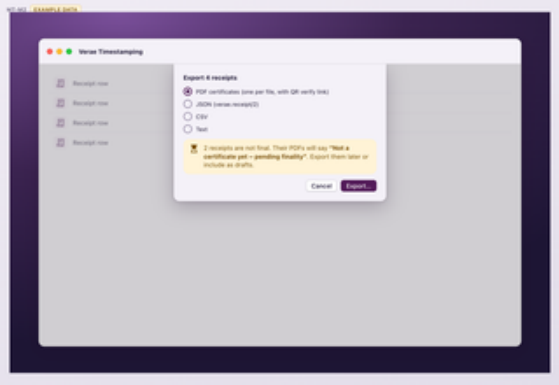
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

MT-M2 — Export sheet with not-final guard

ExportPDF.swift redesign: certificates are issued only for final receipts; non-final ones export as clearly-marked drafts.

macos



MT-M2

Features	
• Format choice (existing JSON/CSV/Text/PDF).	• Guard against presenting non-final receipts as certificates.

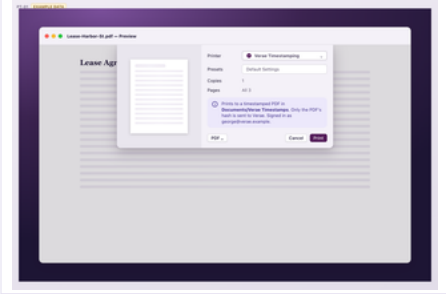
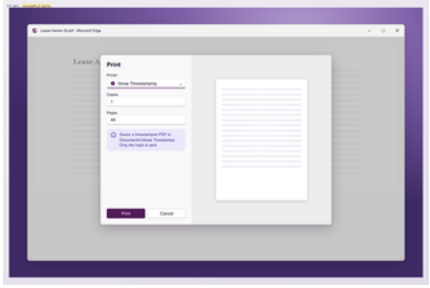
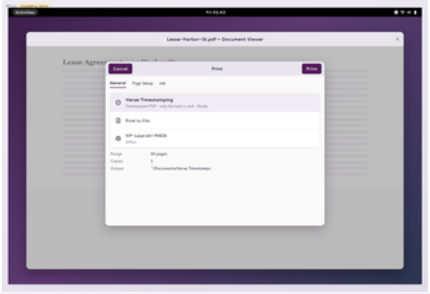
Function / source references
`ExportPDF.swift`
`ContentView.swift exportResults`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-01 — Print dialog → “Verae Timestamping” printer

Entry point: the user prints from any app and picks the Verae Timestamping printer (CUPS backend / IPP). The dialog explains what happens.

macos	windows	debian
		
PT-01	PT-W1	PT-L1

Features	
• Printer named 'Verae Timestamping' (macOS CUPS/IPP queue installed by the .pkg).	• Proposed: printer-info / PPD description text so the system dialog shows what the printer does and who is signed in (via the printer's location/info field).
• Printer installed by install-printer.ps1 (IPP Class Driver).	• Printer comment/location text explains behaviour.
• CUPS queue with descriptive printer-info.	• Output location shown.

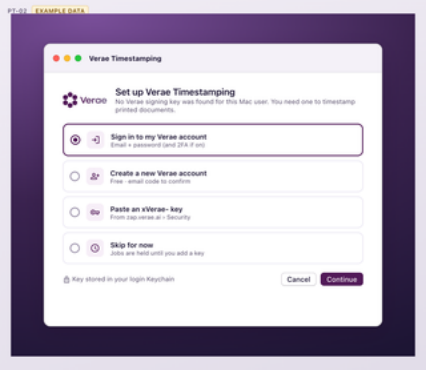
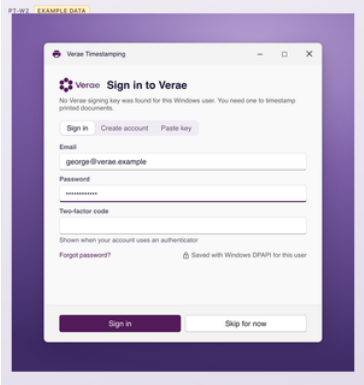
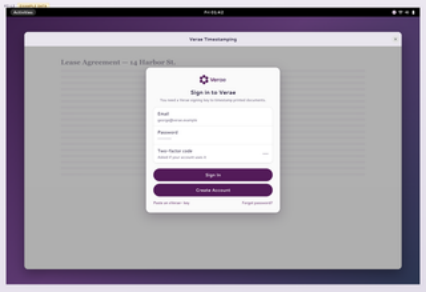
Function / source references
`PrintToVeraeTimeStamp/platform/macos/cups/backend-verae.sh, platform/macos/ipp/install-ipp-printer.sh, cmd/verae-print/main.go`
`PrintToVeraeTimeStamp/platform/windows/scripts/install-printer.ps1`
`internal/winprint/ipp.go, watch.go`
`PrintToVeraeTimeStamp/platform/linux/cups/backend-verae.sh`
`platform/linux/deb/build-deb.sh`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-02 — Sign-in prompt — choose how to connect

Native SwiftUI window replacing the chain of AppleScript dialogs in keyflow_gui.go. Same options, one window, keyboard-navigable.

macos	windows	debian
		
PT-02	PT-W2	PT-L2

Features	
• Why this appears, in one sentence (existing intro copy).	• The five existing options as a radio card list (Keep existing key appears when one exists).
• Where the key is stored (Keychain + ~/.verae today).	• SelectorBar tabs replace the option list.
• 2FA field appears when required.	• Storage note (user-token.ps1 / DPAPI).
• AdwDialog with AdwEntryRow / AdwPasswordEntryRow.	• 2FA row appears when needed.
• Paste key / reset as secondary links.	

Function / source references
`PrintToVeraeTimeStamp/cmd/verae-print/keyflow_gui.go (optKeep/optLogin/optCreate/optPaste/optSkip), keysetup_darwin.go`
`PrintToVeraeTimeStamp/cmd/verae-print/keydialog_windows.go`
`platform/windows/scripts/user-token.ps1`
`PrintToVeraeTimeStamp/cmd/verae-print/keyflow_gui.go (zenity), keysetup_linux.go`

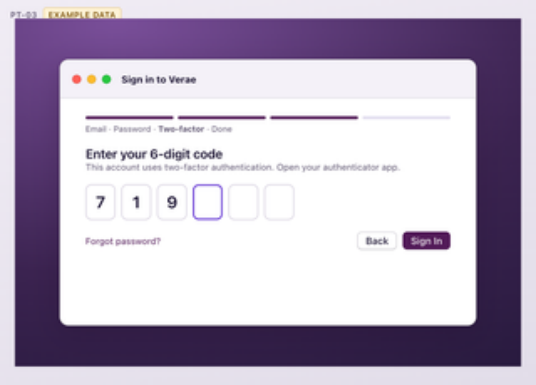
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`portal.2fa`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/2fa/*`	setup:POST {}, code:string, totpCode:string
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-03 — Sign in · 2FA · password reset

The sign-in sequence (email → password → 2FA) and in-app password reset (v0.3.4) as steps in one window.

macos



PT-03

Features	
• Stepper replaces separate modal dialogs.	• Segmented 6-digit code; paste fills all.
• Forgot password → emailed code reset (variant).	

Function / source references

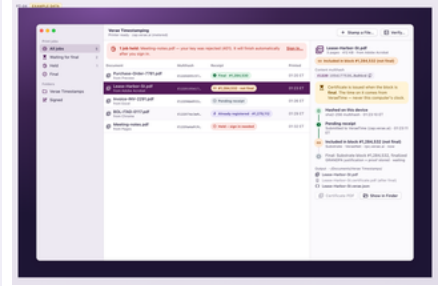
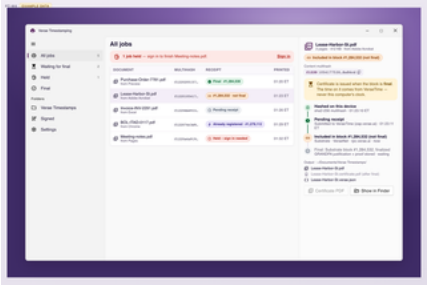
`PrintToVeraeTimeStamp/cmd/verae-print/keyflow_gui.go (sign in, 2FA, reset flow), internal/auth/portal_reset.go`

API id	Method	Path	Key params (name:type)
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`portal.2fa`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/2fa/*`	setup:POST {}, code:string, totpCode:string
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-04 — Print job queue + receipt status

New window listing every printed job with its receipt state — today success/pending are only visible as files appearing in a folder (U-w3).

macos	windows	debian
		
PT-04	PT-W4	PT-L3

Features	
• Held-job banner: 401/403 jobs are kept in the pending folder and retried after a new key (pending.go).	• Job table: document, source app, multihash, receipt state. A pending zap response is held as 'Pending receipt' — never failed and never shown as a jobId (PR2).
• Detail: state, multihash, timeline, output files; certificate only after final; never uses the local clock (PR1).	• InfoBar for held jobs; same table/detail components; settings page mirrors PT-07 (route, output folder, Signed/).
• Held-job banner.	• Notification on final / held with actions.

Function / source references
`PrintToVeraeTimeStamp/cmd/verae-print/pending.go, stampcore.go, serve.go`
`internal/job/job.go`
`internal/veraeapi/client.go`
`PrintToVeraeTimeStamp/internal/job/job.go`
`PrintToVeraeTimeStamp/platform/linux/README.txt`

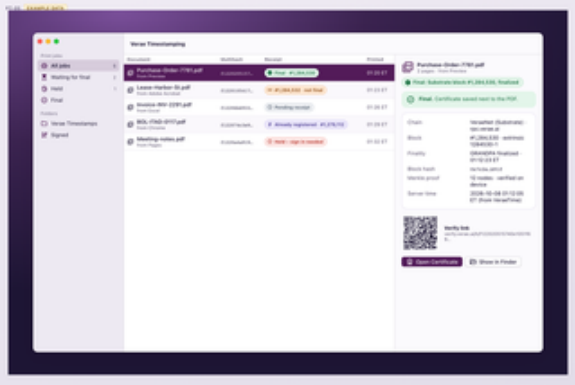
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-05 — Final receipt + certificate ready

Job detail in the final state: proof summary, verify link and QR, and the certificate.

macos



PT-05

Features	
• Final state with green banner.	• Proof: chain, block, extrinsic, GRANDPA finality, Merkle proof, server time.
• Verify link + QR (same as certificate).	

Function / source references

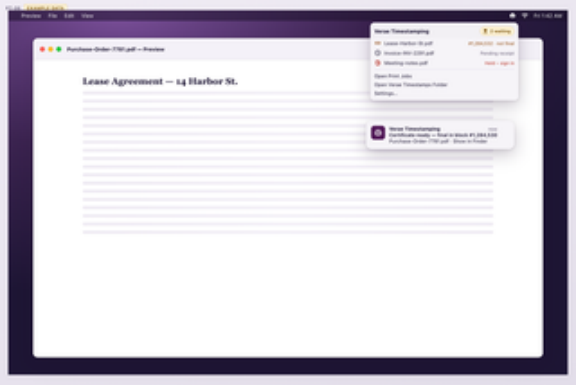
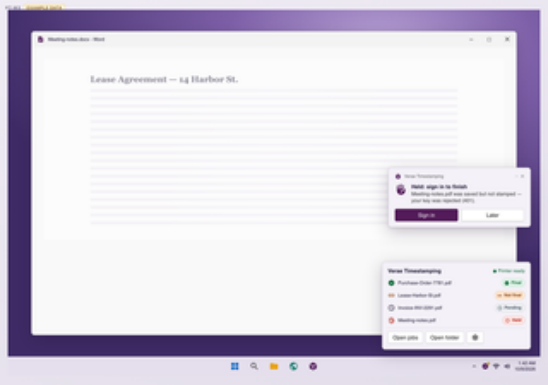
`PrintToVeraeTimeStamp/internal/pdfcert/pdfcert.go (QR LookupURL → explorer.verae.ai/time)`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-06 — Menu bar status + notifications (Mac)

Ambient job status after printing: menu bar extra with waiting/held jobs; notifications for 'certificate ready' and 'held'.

macos	windows
	
PT-06	PT-W3

Features	
• Menu bar extra: waiting / held jobs with stage text.	• Notification when final (certificate ready) and when held (sign in).
• Tray flyout: printer health + recent jobs with state badges.	• Toast for held job with Sign in action (also 'Certificate ready' toast on final).

Function / source references
`New (PR3)`
`PrintToVeraeTimeStamp/cmd/verae-print/serve.go, hide_windows.go, platform_windows.go`
`lib/files/file_browser.dart`
`lib/timestamps/auto_stamp.dart`

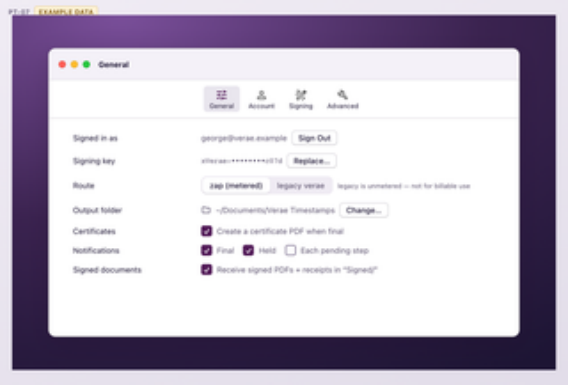
API id	Method	Path	Key params (name:type)
`peergos.cube`	—	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-07 — Settings (Mac)

One settings window replacing scattered config: account/key, route, output folder, certificate and notification options, Signed/ folder subscription (doc 23 ES4).

macos



PT-07

Features	
• Account + key (masked) with replace.	• Route defaults to zap (metered); the macOS CUPS script's legacy 'verae' default is removed (doc 22 §2.5).
• Signed/ folder subscription for the signing round-trip.	

Function / source references
`PrintToVeraeTimeStamp/internal/settings/settings.go`
`platform/macos/cups/backend-verae.sh (channel default)`
`lib/files/file_browser.dart`
`lib/timestamps/auto_stamp.dart`

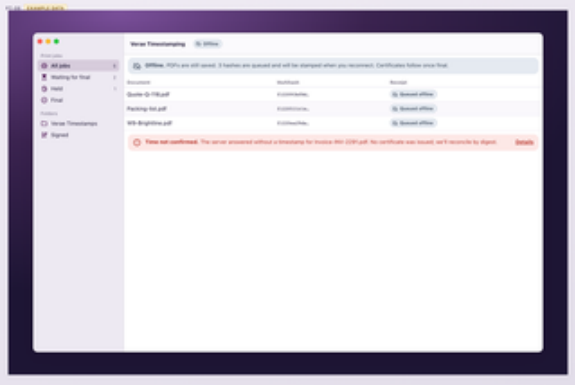
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-08 — Offline & server-error states (printit)

Shows queued stamps offline and the 'time not confirmed' error that replaces today's local-clock fallback (U-w2/PR1).

macos



PT-08

Features	
• Offline banner; PDFs still written, stamps queued.	• Server omitted time → no certificate, reconcile by digest (never local clock).

Function / source references

`PrintToVeraeTimeStamp/internal/veraeapi/client.go (≈330 local-clock fallback — removed)`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-A1 — Android PrintService “Verae Timestamping” (proposal)

Android supports print-service plugins, so a true 'print to Verae' is possible: the Verae app registers a PrintService that appears in the system print dialog.

android

PT-A1

Features	
• Verae Timestamping as a printer in the Android print UI (PrintService).	• Explains where results appear.

Function / source references

New

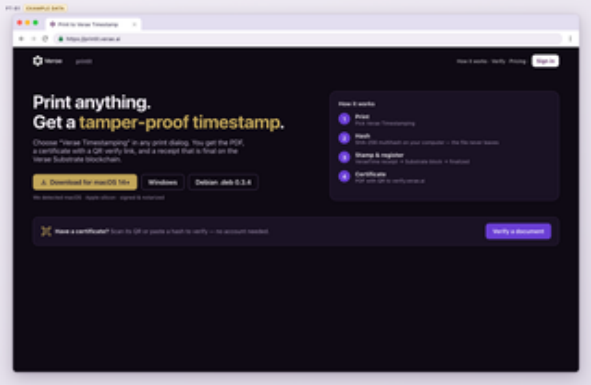
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-B1 — printit.verae.ai — download & how it works

Refresh of the existing dark site: OS-detected download, a 4-step explainer that uses the final-receipt language, and a verify entry.

web



PT-B1

Features	
• Hero copy: certificate + QR + final on Substrate.	• OS-detected primary download; other platforms visible.
• 4-step explainer.	• Verify entry → ZP-V1.

Function / source references
`PrintToVeraeTimeStamp/site/printit.verae.ai/index.html (palette var(--night) / var(--gold) / var(--violet))`

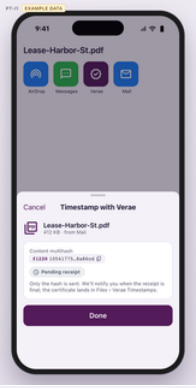
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-I1 — iOS share extension “Timestamp with Verae” (proposal)

printit has no mobile client. iOS cannot add virtual printers, so the equivalent is a share/action extension (from Mail, Files, Print preview ■ Share) that hashes and stamps.

ios



PT-I1

Features	
• Extension shows the file and computes the hash on device.	• Pending state + where the certificate will land; finality via push.

Function / source references

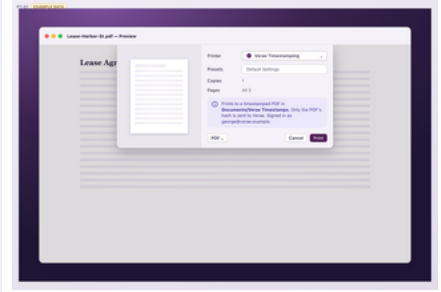
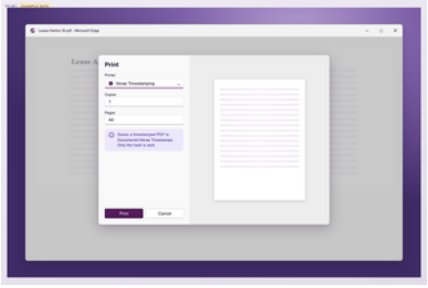
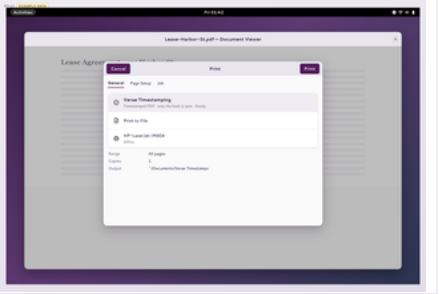
`New (ES4 / doc 22 'no mobile')`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-L1 — Print dialog → “Verae Timestamping” printer

Entry point: the user prints from any app and picks the Verae Timestamping printer (CUPS backend / IPP). The dialog explains what happens.

macos	windows	debian
		
PT-01	PT-W1	PT-L1

Features	
• Printer named 'Verae Timestamping' (macOS CUPS/IPP queue installed by the .pkg).	• Proposed: printer-info / PPD description text so the system dialog shows what the printer does and who is signed in (via the printer's location/info field).
• Printer installed by install-printer.ps1 (IPP Class Driver).	• Printer comment/location text explains behaviour.
• CUPS queue with descriptive printer-info.	• Output location shown.

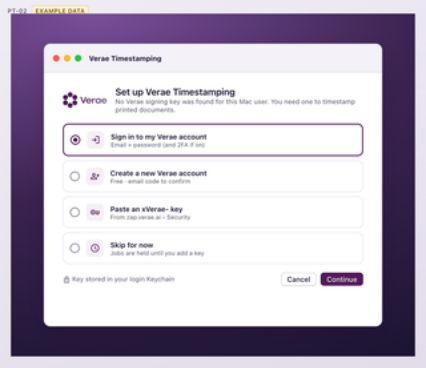
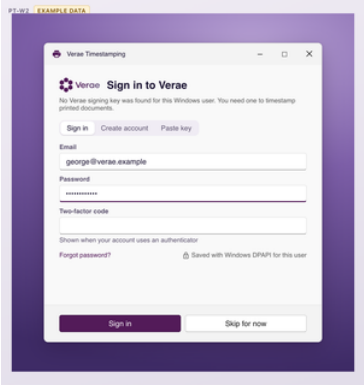
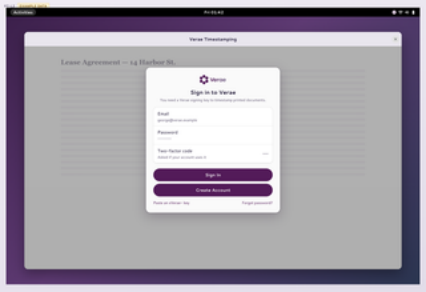
Function / source references
`PrintToVeraeTimeStamp/platform/macos/cups/backend-verae.sh, platform/macos/ipp/install-ipp-printer.sh, cmd/verae-print/main.go`
`PrintToVeraeTimeStamp/platform/windows/scripts/install-printer.ps1`
`internal/winprint/ipp.go, watch.go`
`PrintToVeraeTimeStamp/platform/linux/cups/backend-verae.sh`
`platform/linux/deb/build-deb.sh`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-L2 — Sign-in prompt — choose how to connect

Native SwiftUI window replacing the chain of AppleScript dialogs in keyflow_gui.go. Same options, one window, keyboard-navigable.

macos	windows	debian
		
PT-02	PT-W2	PT-L2

Features	
• Why this appears, in one sentence (existing intro copy).	• The five existing options as a radio card list (Keep existing key appears when one exists).
• Where the key is stored (Keychain + ~/.verae today).	• SelectorBar tabs replace the option list.
• 2FA field appears when required.	• Storage note (user-token.ps1 / DPAPI).
• AdwDialog with AdwEntryRow / AdwPasswordEntryRow.	• 2FA row appears when needed.
• Paste key / reset as secondary links.	

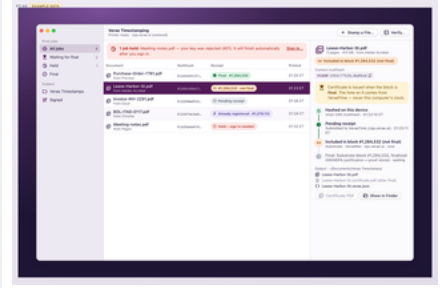
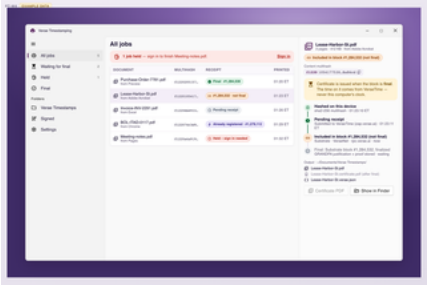
Function / source references
`PrintToVeraeTimeStamp/cmd/verae-print/keyflow_gui.go (optKeep/optLogin/optCreate/optPaste/optSkip), keysetup_darwin.go`
`PrintToVeraeTimeStamp/cmd/verae-print/keydialog_windows.go`
`platform/windows/scripts/user-token.ps1`
`PrintToVeraeTimeStamp/cmd/verae-print/keyflow_gui.go (zenity), keysetup_linux.go`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`portal.2fa`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/2fa/*`	setup:POST {}, code:string, totpCode:string
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-L3 — Print job queue + receipt status

New window listing every printed job with its receipt state — today success/pending are only visible as files appearing in a folder (U-w3).

macos	windows	debian
		
PT-04	PT-W4	PT-L3

Features	
• Held-job banner: 401/403 jobs are kept in the pending folder and retried after a new key (pending.go).	• Job table: document, source app, multihash, receipt state. A pending zap response is held as 'Pending receipt' — never failed and never shown as a jobId (PR2).
• Detail: state, multihash, timeline, output files; certificate only after final; never uses the local clock (PR1).	• InfoBar for held jobs; same table/detail components; settings page mirrors PT-07 (route, output folder, Signed/).
• Held-job banner.	• Notification on final / held with actions.

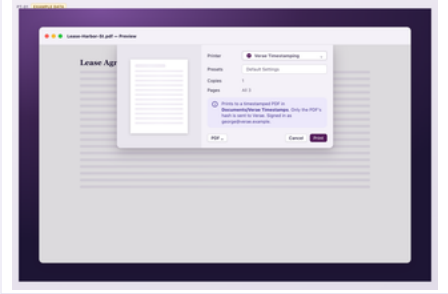
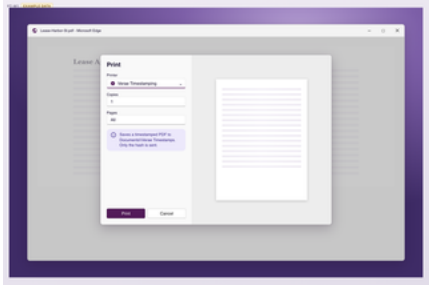
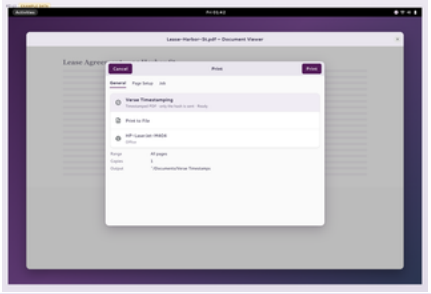
Function / source references
`PrintToVeraeTimeStamp/cmd/verae-print/pending.go, stampcore.go, serve.go`
`internal/job/job.go`
`internal/veraeapi/client.go`
`PrintToVeraeTimeStamp/internal/job/job.go`
`PrintToVeraeTimeStamp/platform/linux/README.txt`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-W1 — Print dialog → “Verae Timestamping” printer

Entry point: the user prints from any app and picks the Verae Timestamping printer (CUPS backend / IPP). The dialog explains what happens.

macos	windows	debian
		
PT-01	PT-W1	PT-L1

Features	
• Printer named 'Verae Timestamping' (macOS CUPS/IPP queue installed by the .pkg).	• Proposed: printer-info / PPD description text so the system dialog shows what the printer does and who is signed in (via the printer's location/info field).
• Printer installed by install-printer.ps1 (IPP Class Driver).	• Printer comment/location text explains behaviour.
• CUPS queue with descriptive printer-info.	• Output location shown.

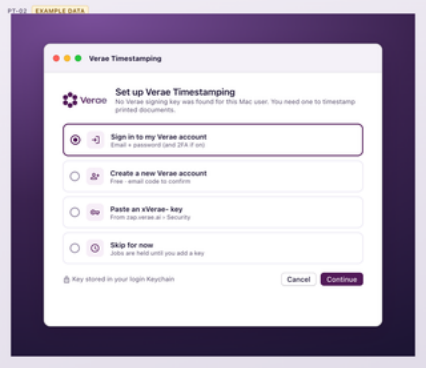
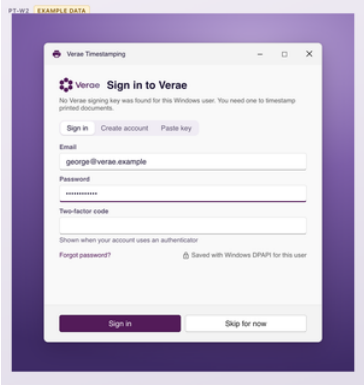
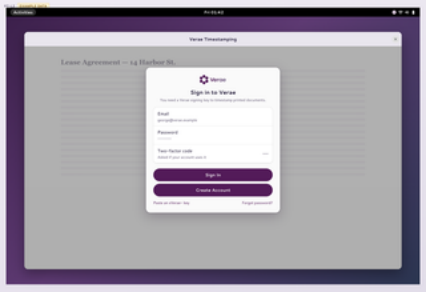
Function / source references
`PrintToVeraeTimeStamp/platform/macos/cups/backend-verae.sh, platform/macos/ipp/install-ipp-printer.sh, cmd/verae-print/main.go`
`PrintToVeraeTimeStamp/platform/windows/scripts/install-printer.ps1`
`internal/winprint/ipp.go, watch.go`
`PrintToVeraeTimeStamp/platform/linux/cups/backend-verae.sh`
`platform/linux/deb/build-deb.sh`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-W2 — Sign-in prompt — choose how to connect

Native SwiftUI window replacing the chain of AppleScript dialogs in keyflow_gui.go. Same options, one window, keyboard-navigable.

macos	windows	debian
		
PT-02	PT-W2	PT-L2

Features	
• Why this appears, in one sentence (existing intro copy).	• The five existing options as a radio card list (Keep existing key appears when one exists).
• Where the key is stored (Keychain + ~/.verae today).	• SelectorBar tabs replace the option list.
• 2FA field appears when required.	• Storage note (user-token.ps1 / DPAPI).
• AdwDialog with AdwEntryRow / AdwPasswordEntryRow.	• 2FA row appears when needed.
• Paste key / reset as secondary links.	

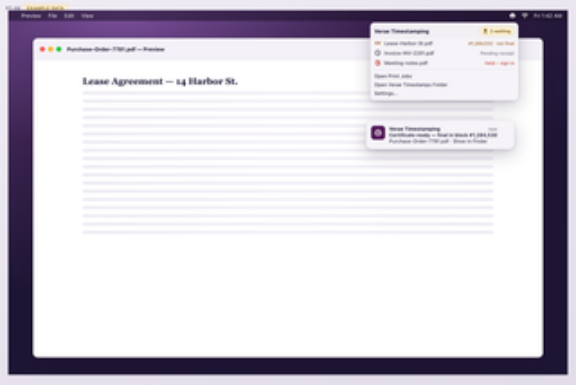
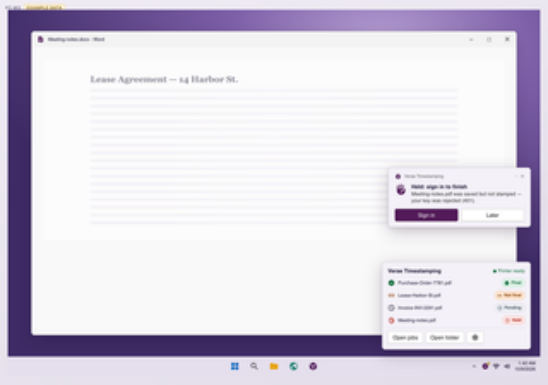
Function / source references
`PrintToVeraeTimeStamp/cmd/verae-print/keyflow_gui.go (optKeep/optLogin/optCreate/optPaste/optSkip), keysetup_darwin.go`
`PrintToVeraeTimeStamp/cmd/verae-print/keydialog_windows.go`
`platform/windows/scripts/user-token.ps1`
`PrintToVeraeTimeStamp/cmd/verae-print/keyflow_gui.go (zenity), keysetup_linux.go`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`portal.2fa`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/2fa/*`	setup:POST {}, code:string, totpCode:string
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-W3 — Menu bar status + notifications (Mac)

Ambient job status after printing: menu bar extra with waiting/held jobs; notifications for 'certificate ready' and 'held'.

macos	windows
	
PT-06	PT-W3

Features	
• Menu bar extra: waiting / held jobs with stage text.	• Notification when final (certificate ready) and when held (sign in).
• Tray flyout: printer health + recent jobs with state badges.	• Toast for held job with Sign in action (also 'Certificate ready' toast on final).

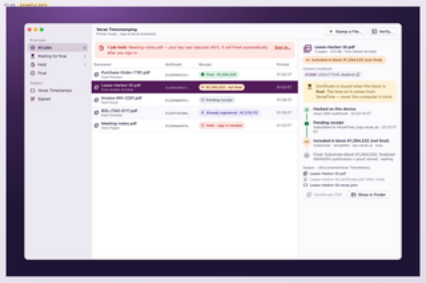
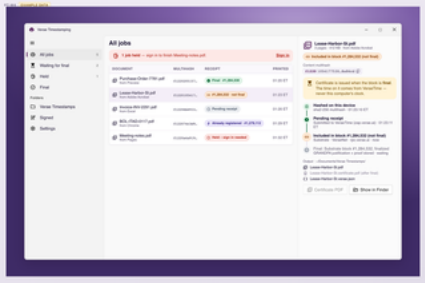
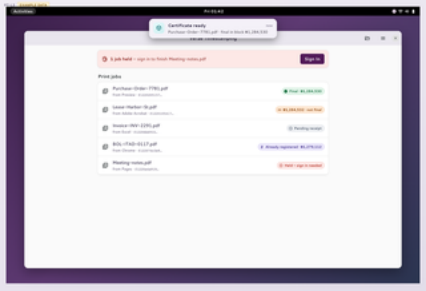
Function / source references
`New (PR3)`
`PrintToVeraeTimeStamp/cmd/verae-print/serve.go, hide_windows.go, platform_windows.go`
`lib/files/file_browser.dart`
`lib/timestamps/auto_stamp.dart`

API id	Method	Path	Key params (name:type)
`peergos.cube`	—	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

PT-W4 — Print job queue + receipt status

New window listing every printed job with its receipt state — today success/pending are only visible as files appearing in a folder (U-w3).

macos	windows	debian
		
PT-04	PT-W4	PT-L3

Features	
• Held-job banner: 401/403 jobs are kept in the pending folder and retried after a new key (pending.go).	• Job table: document, source app, multihash, receipt state. A pending zap response is held as 'Pending receipt' — never failed and never shown as a jobId (PR2).
• Detail: state, multihash, timeline, output files; certificate only after final; never uses the local clock (PR1).	• InfoBar for held jobs; same table/detail components; settings page mirrors PT-07 (route, output folder, Signed/).
• Held-job banner.	• Notification on final / held with actions.

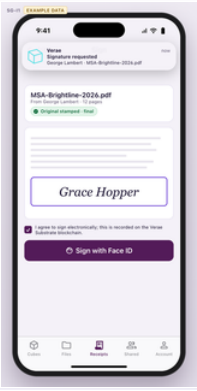
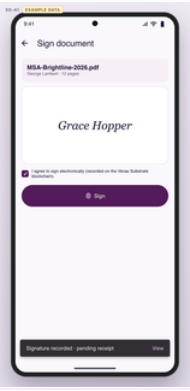
Function / source references
`PrintToVeraeTimeStamp/cmd/verae-print/pending.go, stampcore.go, serve.go`
`internal/job/job.go`
`internal/veraeapi/client.go`
`PrintToVeraeTimeStamp/internal/job/job.go`
`PrintToVeraeTimeStamp/platform/linux/README.txt`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

SG-A1 — iOS — signature request + sign

Push notification opens the native sign screen; Face ID confirms; the signature event is stamped.

ios	android
	
SG-I1	SG-A1

Features	
• Push notification (APNs).	• Document card with original's final receipt.
• Signature field (PencilKit draw / type).	• Consent + Face ID confirmation.
• Document card.	• Signature canvas (stylus/finger).

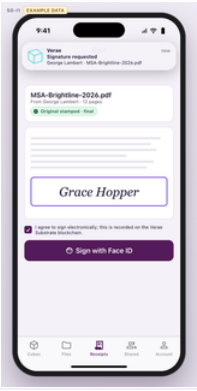
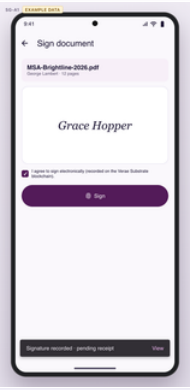
Function / source references
`verae-sign /sign/v1 on sign.verae.ai (edge)`
`stamps via /v1/timestamp on customer.verae.ai — not Zapier`
`digests only`
`peergos-flutter app shell`
`lib/sign/sign_api.dart`
`lib/sign/sign_screens.dart`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

SG-I1 — iOS — signature request + sign

Push notification opens the native sign screen; Face ID confirms; the signature event is stamped.

ios	android
	
SG-I1	SG-A1

Features	
• Push notification (APNs).	• Document card with original's final receipt.
• Signature field (PencilKit draw / type).	• Consent + Face ID confirmation.
• Document card.	• Signature canvas (stylus/finger).

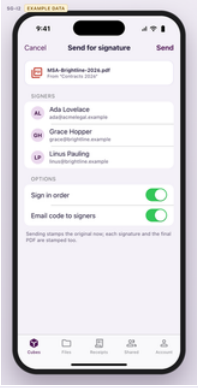
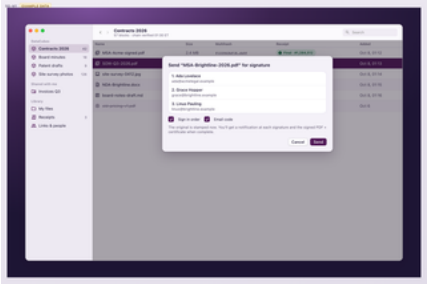
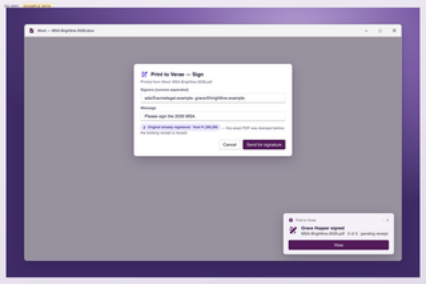
Function / source references
`verae-sign /sign/v1 on sign.verae.ai (edge)`
`stamps via /v1/timestamp on customer.verae.ai — not Zapier`
`digests only`
`peergos-flutter app shell`
`lib/sign/sign_api.dart`
`lib/sign/sign_screens.dart`

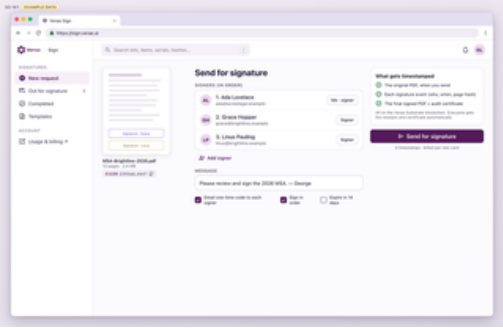
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

SG-I2 — iOS — send for signature (from a cube file)

Start a request from a file in a DataCube (share menu → Send for signature).

ios	macos	windows
		
SG-I2	SG-M1	SG-WN1

web

SG-W1

Features	
• Signer list from contacts.	• What-gets-stamped note.
• Signers.	• Stamping explanation; notifications per step.
• Signer entry from print dialog.	• Already-registered original reuses receipt.
• Document with placed signature fields and its multihash.	• Ordered signers.
• Email OTP + ordering options.	• Plain explanation of what is stamped (send, each signature, completion).

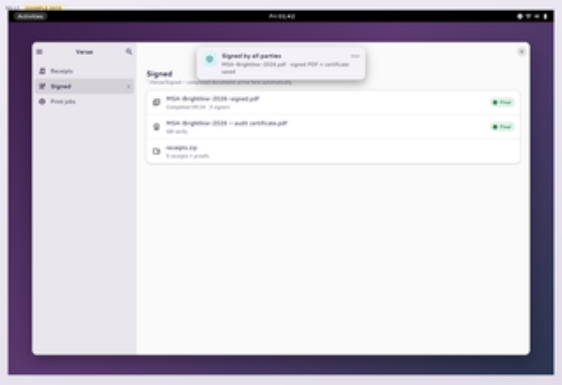
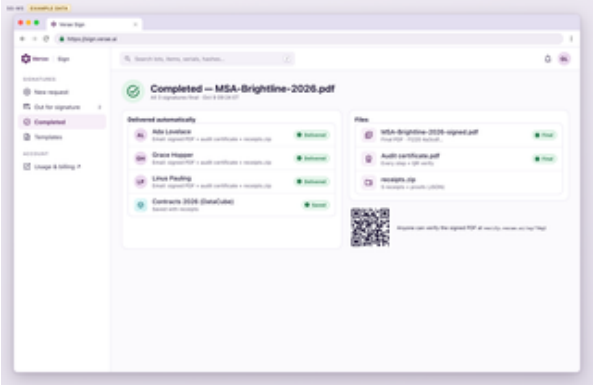
Function / source references
`verae-sign /sign/v1 on sign.verae.ai (edge)`
`stamps via /v1/timestamp on customer.verae.ai — not Zapier`
`digests only`
`printit Windows print server`
`lib/sign/sign_api.dart`
`lib/sign/sign_screens.dart`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

SG-L1 — Debian GNOME — completion notification + Signed folder

Linux helper shows a GNOME notification on completion and drops files into ~/Verae/Signed.

debian	web
	
SG-L1	SG-W5

Features	
• Signed folder listing with receipt states.	• Notification (libnotify) with action.
• Delivery fan-out per party + DataCube save.	• Files with final receipts.
• Verify QR/link.	

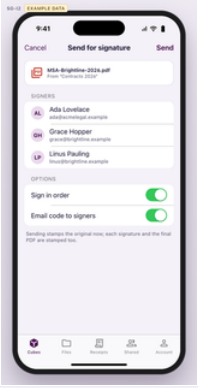
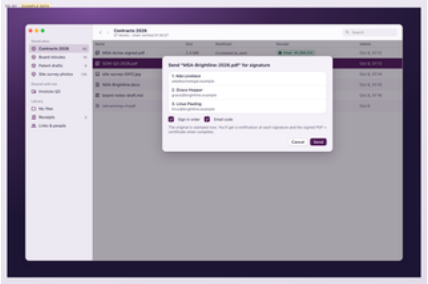
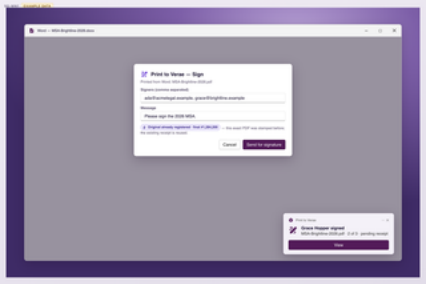
Function / source references
`verae-sign /sign/v1 on sign.verae.ai (edge)`
`stamps via /v1/timestamp on customer.verae.ai — not Zapier`
`digests only`
`printit Linux CUPS helper`
`lib/sign/sign_api.dart`
`lib/sign/sign_screens.dart`

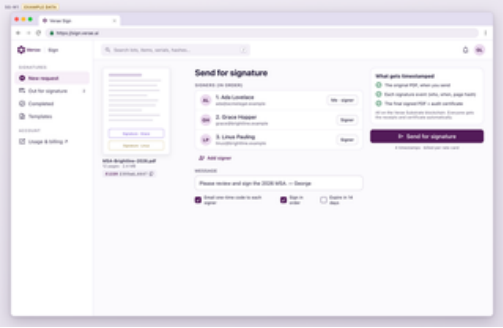
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

SG-M1 — iOS — send for signature (from a cube file)

Start a request from a file in a DataCube (share menu → Send for signature).

ios	macos	windows
		
SG-I2	SG-M1	SG-WN1

web

SG-W1

Features	
• Signer list from contacts.	• What-gets-stamped note.
• Signers.	• Stamping explanation; notifications per step.
• Signer entry from print dialog.	• Already-registered original reuses receipt.
• Document with placed signature fields and its multihash.	• Ordered signers.
• Email OTP + ordering options.	• Plain explanation of what is stamped (send, each signature, completion).

Function / source references
`verae-sign /sign/v1 on sign.verae.ai (edge)`
`stamps via /v1/timestamp on customer.verae.ai — not Zapier`
`digests only`
`printit Windows print server`
`lib/sign/sign_api.dart`
`lib/sign/sign_screens.dart`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

SG-P1 — Signature audit certificate (PDF)

Certificate delivered with the signed PDF: hashes, every step with its Substrate block, verify QR.

paper

SG-P1

Features	
• Original + signed hashes and overall state.	• Per-step table with block numbers.
• Verify QR + instructions.	

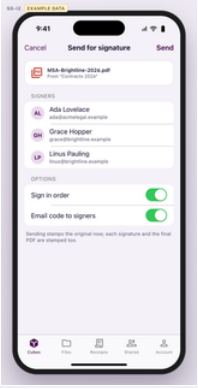
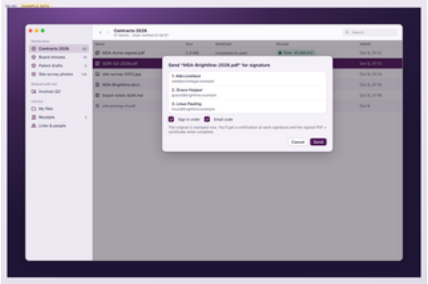
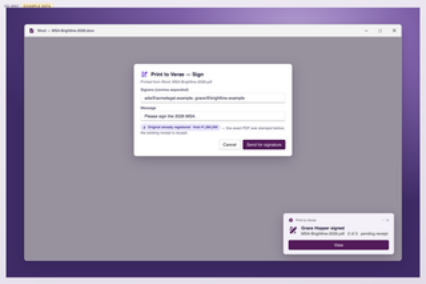
Function / source references
`verae-sign /sign/v1 on sign.verae.ai (edge)`
`stamps via /v1/timestamp on customer.verae.ai — not Zapier`
`digests only`
`printit/pdfcert.go layout`
`lib/sign/sign_api.dart`
`lib/sign/sign_screens.dart`

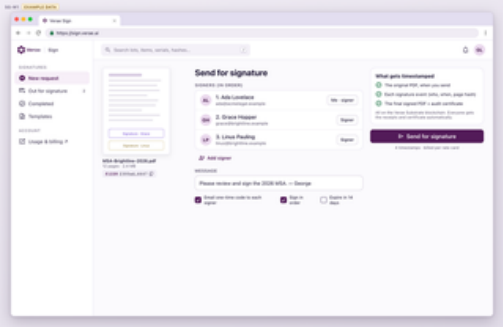
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

SG-W1 — iOS — send for signature (from a cube file)

Start a request from a file in a DataCube (share menu → Send for signature).

ios	macos	windows
		
SG-I2	SG-M1	SG-WN1

web

SG-W1

Features	
• Signer list from contacts.	• What-gets-stamped note.
• Signers.	• Stamping explanation; notifications per step.
• Signer entry from print dialog.	• Already-registered original reuses receipt.
• Document with placed signature fields and its multihash.	• Ordered signers.
• Email OTP + ordering options.	• Plain explanation of what is stamped (send, each signature, completion).

Function / source references
`verae-sign /sign/v1 on sign.verae.ai (edge)`
`stamps via /v1/timestamp on customer.verae.ai — not Zapier`
`digests only`
`printit Windows print server`
`lib/sign/sign_api.dart`
`lib/sign/sign_screens.dart`

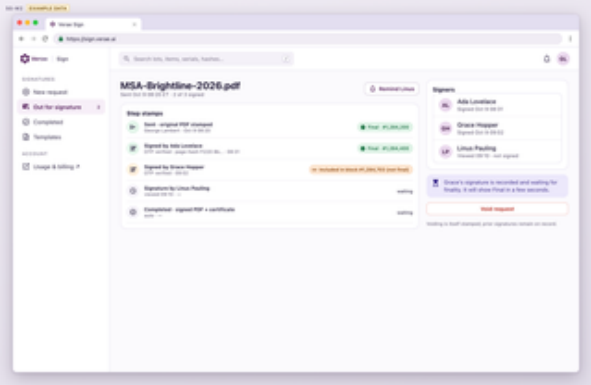
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

SG-W2 — Request status — step stamps

Live progress with one receipt per step; never a job id.

web



SG-W2

Features	
• Step list with Substrate state per step.	• Signer status.
• Pending finality explanation; void is stamped.	

Function / source references
`verae-sign /sign/v1 on sign.verae.ai (edge)`
`stamps via /v1/timestamp on customer.verae.ai — not Zapier`
`digests only`
`lib/sign/sign_api.dart`
`lib/sign/sign_screens.dart`

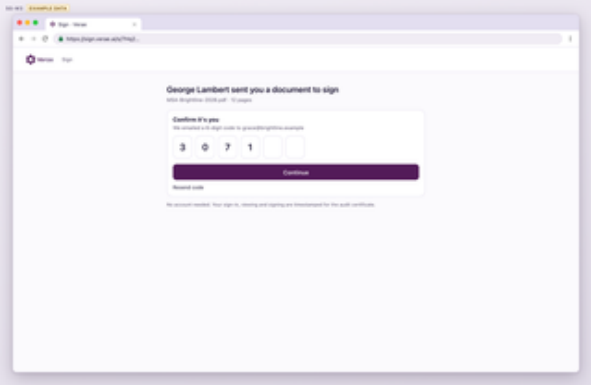
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

SG-W3 — Signer landing + email OTP

Account-free signer entry from the email link; identity proven by a one-time code.

web



SG-W3

Features	
• OTP to the invited address.	• Privacy/audit note.

Function / source references
`verae-sign /sign/v1 on sign.verae.ai (edge)`
`stamps via /v1/timestamp on customer.verae.ai — not Zapier`
`digests only`
`lib/sign/sign_api.dart`
`lib/sign/sign_screens.dart`

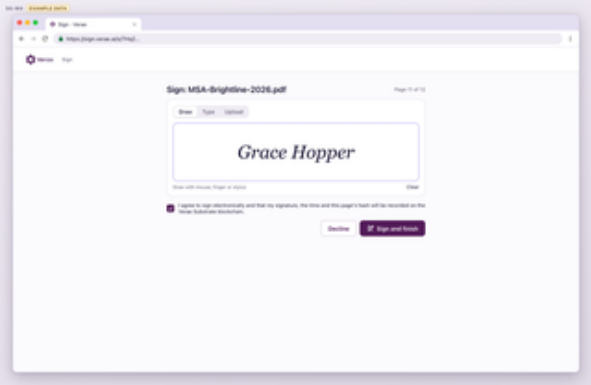
API id	Method	Path	Key params (name:type)
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

SG-W4 — Sign — draw / type / upload + consent

Signature capture with explicit e-sign consent; submitting stamps the signature event.

web



SG-W4

Features	
• Draw/Type/Upload signature.	• Consent text that names the blockchain record.
• Decline (stamped with reason) or Sign.	

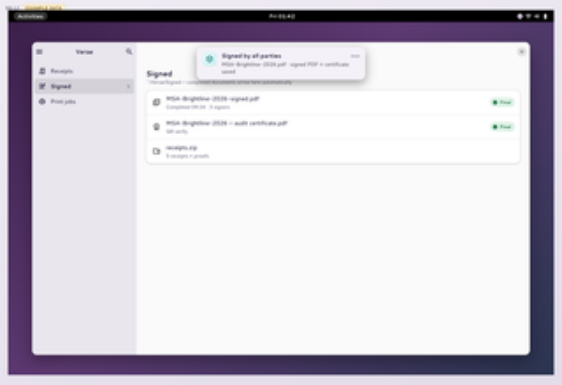
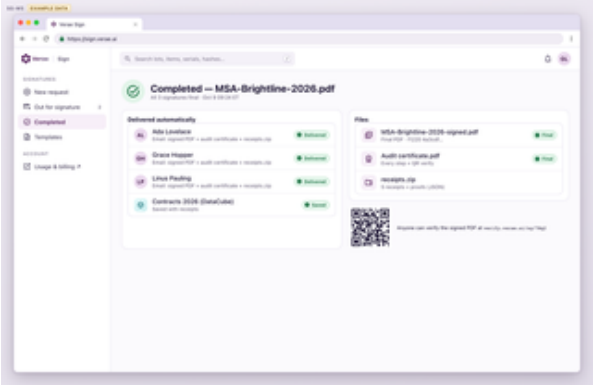
Function / source references
`verae-sign /sign/v1 on sign.verae.ai (edge)`
`stamps via /v1/timestamp on customer.verae.ai — not Zapier`
`digests only`
`lib/sign/sign_api.dart`
`lib/sign/sign_screens.dart`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

SG-W5 — Debian GNOME — completion notification + Signed folder

Linux helper shows a GNOME notification on completion and drops files into ~/Verae/Signed.

debian	web
	
SG-L1	SG-W5

Features	
• Signed folder listing with receipt states.	• Notification (libnotify) with action.
• Delivery fan-out per party + DataCube save.	• Files with final receipts.
• Verify QR/link.	

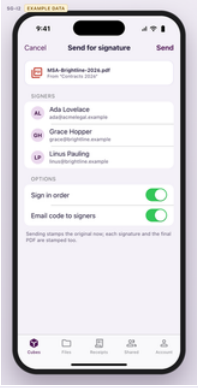
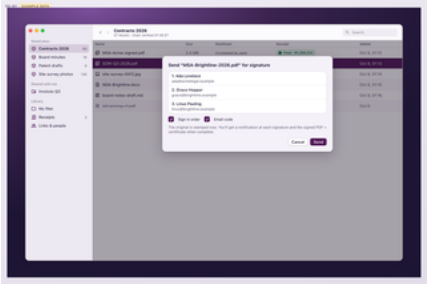
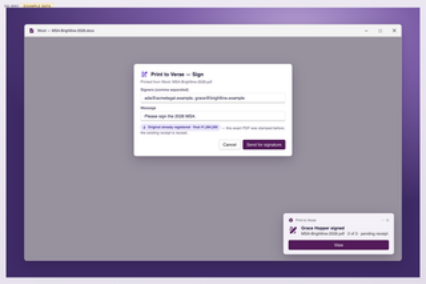
Function / source references
`verae-sign /sign/v1 on sign.verae.ai (edge)`
`stamps via /v1/timestamp on customer.verae.ai — not Zapier`
`digests only`
`printit Linux CUPS helper`
`lib/sign/sign_api.dart`
`lib/sign/sign_screens.dart`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …

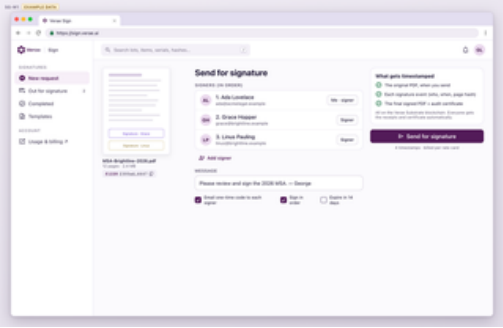
Sample calls + full param tables → VERAE-API-REFERENCE.pdf

SG-WN1 — iOS — send for signature (from a cube file)

Start a request from a file in a DataCube (share menu → Send for signature).

ios	macos	windows
		
SG-I2	SG-M1	SG-WN1

web



SG-W1

Features	
• Signer list from contacts.	• What-gets-stamped note.
• Signers.	• Stamping explanation; notifications per step.
• Signer entry from print dialog.	• Already-registered original reuses receipt.
• Document with placed signature fields and its multihash.	• Ordered signers.
• Email OTP + ordering options.	• Plain explanation of what is stamped (send, each signature, completion).

Function / source references
`verae-sign /sign/v1 on sign.verae.ai (edge)`
`stamps via /v1/timestamp on customer.verae.ai — not Zapier`
`digests only`
`printit Windows print server`
`lib/sign/sign_api.dart`
`lib/sign/sign_screens.dart`

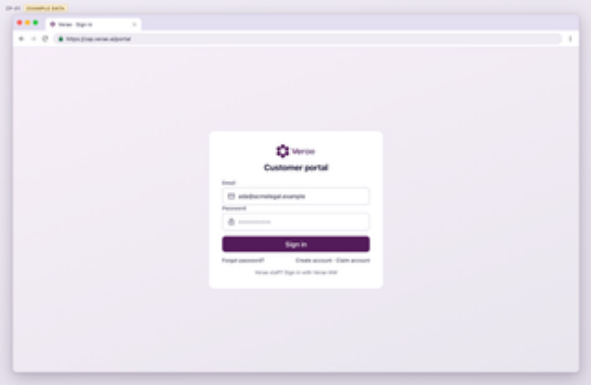
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`sign.requests`	**POST**	`sign.verae.ai/sign/v1/requests`	documentSha256:string 64 hex, documentName:string, signers:array<{email:string,name?:string}>, title:string?, message:string?, …
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`add.destination`	**local + APIs**	`App add flow (DC-07D)`	uploadPeergos:boolean, addTags:boolean, timestampRegistry:boolean, tags:string[], metadataJsonl:string (JSONL), …
`sign.public.sign`	**POST**	`sign.verae.ai/sign/v1/public/:token/sign`	token:path string, email:string, otp:string, signatureSha256:string 64 hex, kind:'draw'|'type'|'upload', …

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-01 — Portal sign-in + 2FA

Customer sign-in (email + password, optional TOTP), sign-up or claim an account created by sales; staff are routed to IAM (portal/index.html #auth-form).

web



ZP-01

Features	
• Rebranded from 'Zappier Portal' to Verae.	• Email/password; links to reset, sign-up and claim; staff → iam.verae.ai. Variant: TOTP step.

Function / source references

`veb/ (verae-edge-billing) portal/index.html (#auth), docs/CUSTOMER-PORTAL.md`

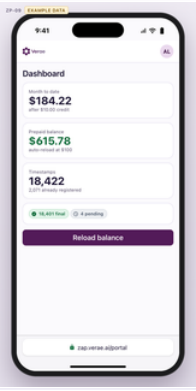
API id	Method	Path	Key params (name:type)
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.2fa`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/2fa/*`	setup:POST {}, code:string, totpCode:string

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-02 — Portal — mobile responsive

Dashboard reflowed for phones: stacked KPIs, bottom tab bar.

web



ZP-09

Features	
• Stacked KPIs (single column under 600 px).	• Receipt health chips.

Function / source references

`veb/ (verae-edge-billing) portal/index.html (viewport meta)`

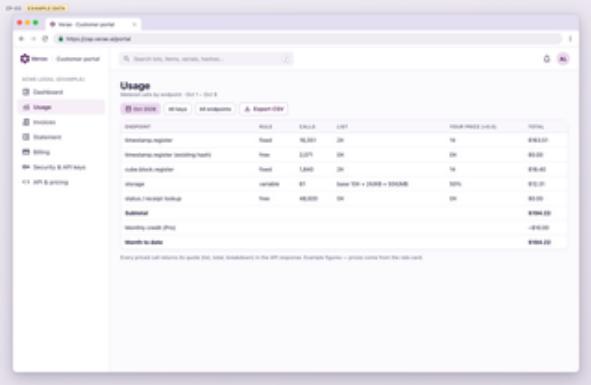
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-03 — Usage & metering detail

Per-endpoint metered usage with the pricing rule, customer-type multiplier and monthly credit (USER-MANUAL §3).

web



ZP-03

Features	
• Period/key/endpoint filters + CSV export.	• Rule → list → multiplier → total; existing-hash lookups are free.
• Quote-per-call explanation.	

Function / source references
`veb/ (verae-edge-billing) docs/USER-MANUAL.md §3`
`src/pricing`

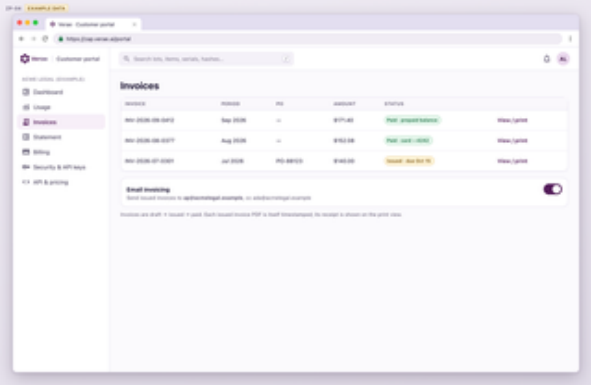
API id	Method	Path	Key params (name:type)
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-04 — Invoices

Invoice history with status, PO number, print view and email-invoicing preferences (portal Invoices tab).

web



ZP-04

Features	
• Invoice table (draft hidden from customers).	• Email invoicing preference.
• Invoice PDF stamped (proposal).	

Function / source references
`veb/ (verae-edge-billing) portal data-tab=invoices`
`docs/ACCOUNTING.md`

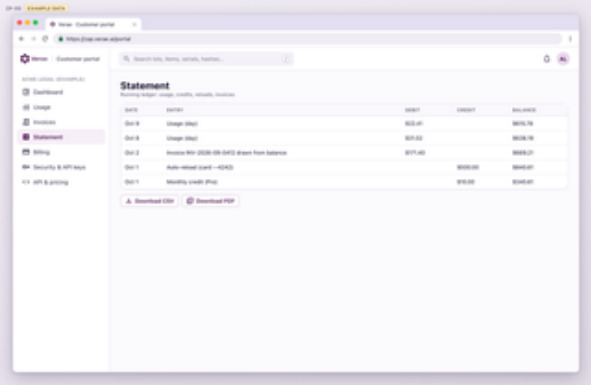
API id	Method	Path	Key params (name:type)
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-05 — Statement (ledger)

Chronological ledger of debits/credits and running balance (portal Statement tab).

web



ZP-05

Features	
• Ledger with running balance.	• Exports.

Function / source references

`veb/ (verae-edge-billing) portal data-tab=statement`

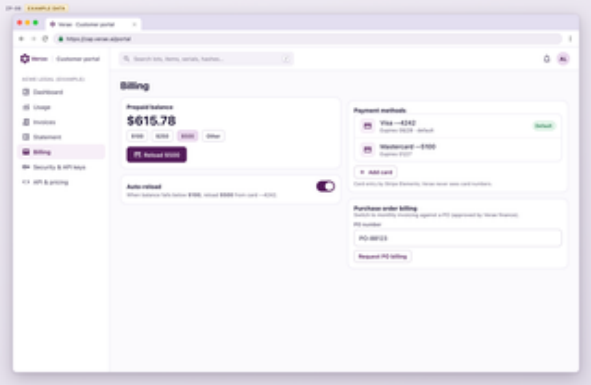
API id	Method	Path	Key params (name:type)
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-06 — Billing — balance, reload, cards, PO

Prepaid balance reloads, auto-reload, Stripe cards and PO billing request (portal Billing tab; flutter usage_billing Cards/PO tabs).

web



ZP-06

Features	
• Balance + reload presets.	• Auto-reload threshold.
• Cards (Stripe Elements).	• PO billing request.

Function / source references
`veb/ (verae-edge-billing) portal data-tab=billing`
`peergos-flutter lib/billing/usage_billing_screen.dart`
`lib/files/file_browser.dart`
`lib/timestamps/auto_stamp.dart`

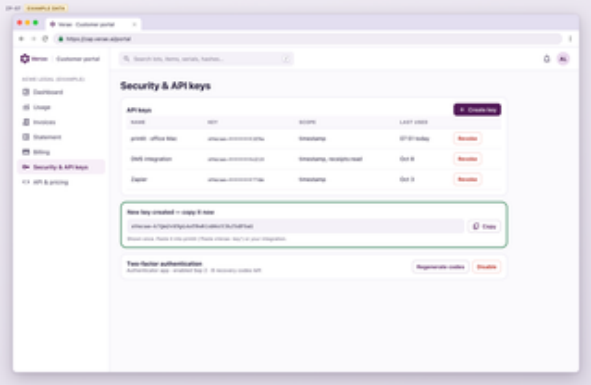
API id	Method	Path	Key params (name:type)
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-07 — Security — API keys (xVerae-) & 2FA

Create/revoke named, scoped API keys (shown once) and manage TOTP 2FA (portal Security tab).

web



ZP-07

Features	
• Key list (masked) with scope, last used, revoke.	• One-time reveal of a new xVerae- key with copy.
• 2FA status and recovery codes.	

Function / source references
`veb/ (verae-edge-billing) portal data-tab=security`
`printit keyflow_gui.go`
`lib/files/file_browser.dart`
`lib/timestamps/auto_stamp.dart`

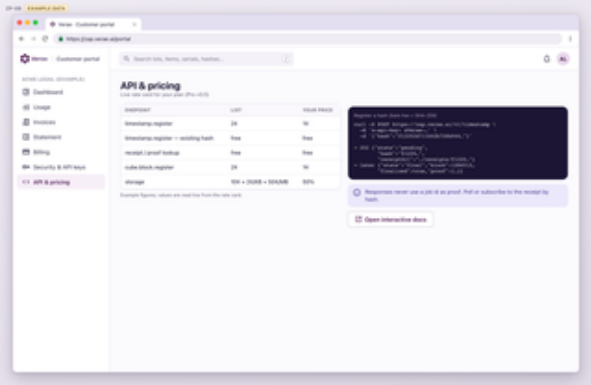
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.2fa`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/2fa/*`	setup:POST {}, code:string, totpCode:string

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-08 — API & pricing

Live per-plan pricing and API quick start; the example teaches the receipt lifecycle by hash, not jobId.

web



ZP-08

Features	
• Rate card at the customer's multiplier.	• Quick-start code with pending → final response.
• No-jobId rule.	

Function / source references
`veb/ (verae-edge-billing) portal data-tab=docs`
`openapi.yaml`

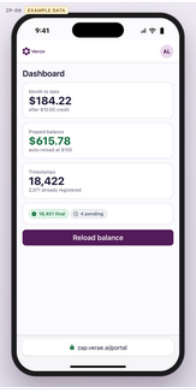
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-09 — Portal — mobile responsive

Dashboard reflowed for phones: stacked KPIs, bottom tab bar.

web



ZP-09

Features	
• Stacked KPIs (single column under 600 px).	• Receipt health chips.

Function / source references

`veb/ (verae-edge-billing) portal/index.html (viewport meta)`

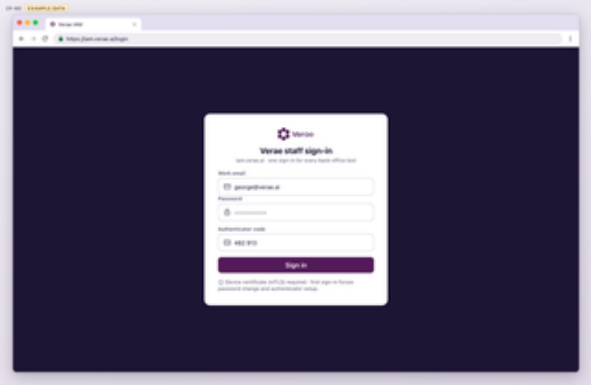
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-B0 — Staff sign-in (Verae IAM)

Single staff sign-in for admin, billing, CS, sales, finance and compliance (verae-staff-iam).

web



ZP-B0

Features	
• IAM-branded staff login (separate from customer portal).	• Mandatory TOTP.
• mTLS device cert + forced password change on first login.	

Function / source references

`verae-staff-iam/public, docs (forcepw, mtls bundles)`

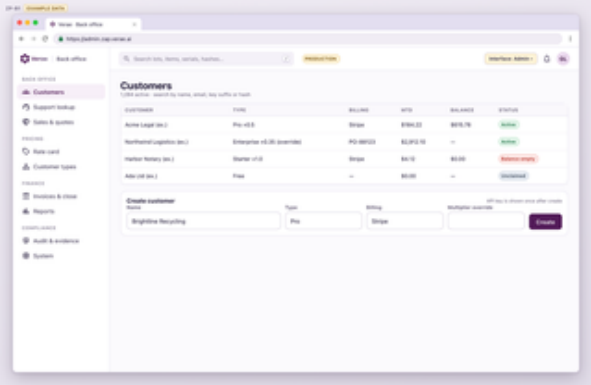
API id	Method	Path	Key params (name:type)
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-B1 — Admin — customers

Customer list with type, billing method, MTD, balance; create customer (key shown once) (admin Customers tab).

web



ZP-B1

Features	
• Customer table with status (balance empty, unclaimed).	• Create with type/billing/override.

Function / source references
`veb/ (verae-edge-billing) admin/index.html data-tab=customers`
`USER-MANUAL §4.3`

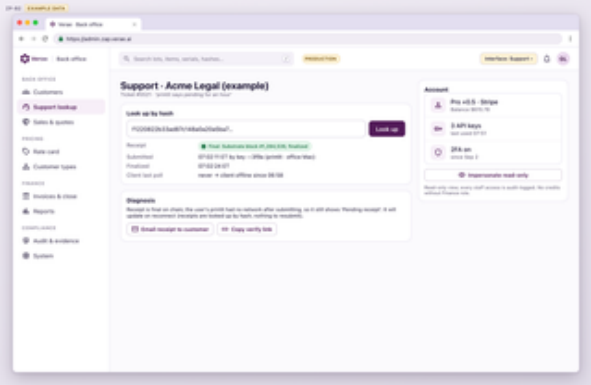
API id	Method	Path	Key params (name:type)
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-B2 — Customer support lookup

CS resolves receipt questions by hash/key/customer; diagnoses client vs chain; read-only impersonation, audit-logged.

web



ZP-B2

Features	
• Hash lookup showing chain truth vs client state.	• Diagnosis + safe actions (email receipt, verify link).
• Account summary; read-only impersonation.	

Function / source references
`proposal`
`veb admin Customers`
`verae-staff-iam roles`

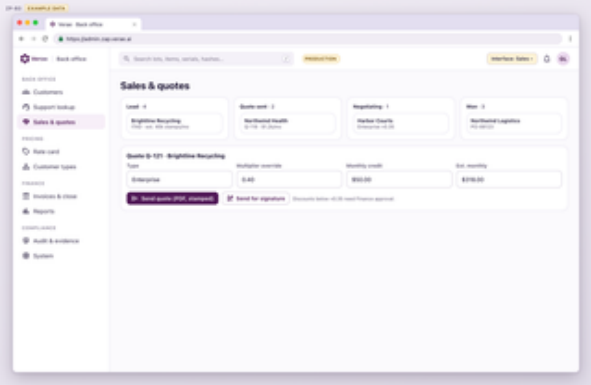
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-B3 — Sales — pipeline & quotes

Pipeline and quote builder using rate-card types/overrides; quotes sent stamped or via signing round-trip.

web



ZP-B3

Features	
• Kanban pipeline.	• Quote builder → stamped PDF or SG signing; approval guardrail.

Function / source references
`proposal`
`veb Customer types / overrides`

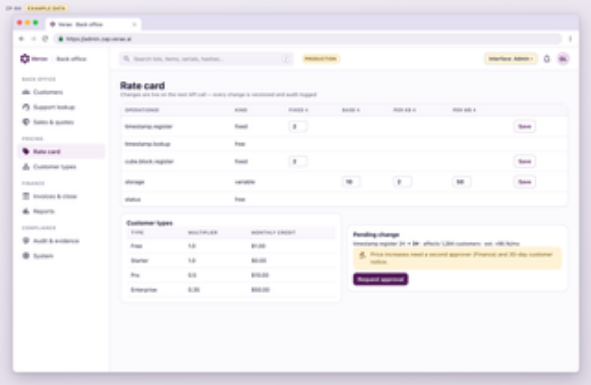
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-B4 — Admin — rate card & customer types

Per-endpoint pricing (free/fixed/variable) and tier multipliers/credits; increases gated by approval (admin Rate card + Customer types tabs).

web



ZP-B4

Features	
• Editable rate card (operationId must match openapi.yaml; delete = endpoint off).	• Types: multiplier + monthly credit.
• Impact preview + 2-person approval (proposal).	

Function / source references
`veb/ (verae-edge-billing) admin data-tab=endpoints, tiers`
`USER-MANUAL §4.1–4.2`

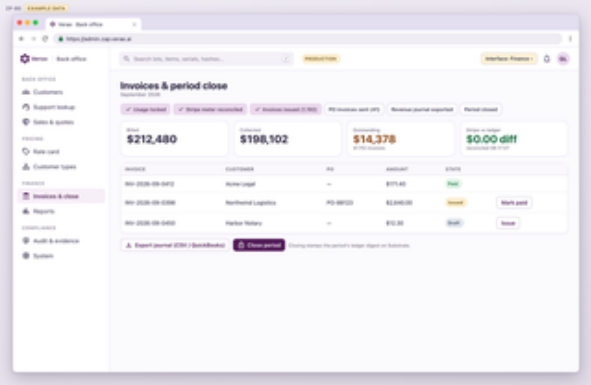
API id	Method	Path	Key params (name:type)
`peergos.cube`	—	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-B5 — Finance — invoices, reconciliation, period close

Month-end checklist: lock usage, reconcile Stripe meter vs ledger, issue invoices (draft→issued→paid), export journal, close period.

web



ZP-B5

Features	
• Close checklist.	• Billed/collected/outstanding/reconciliation KPIs.
• Invoice states with actions.	• Export + close (ledger digest stamped).

Function / source references
`veb/ (verae-edge-billing) admin data-tab=invoices, statement`
`docs/ACCOUNTING.md`
`USER-MANUAL §6`

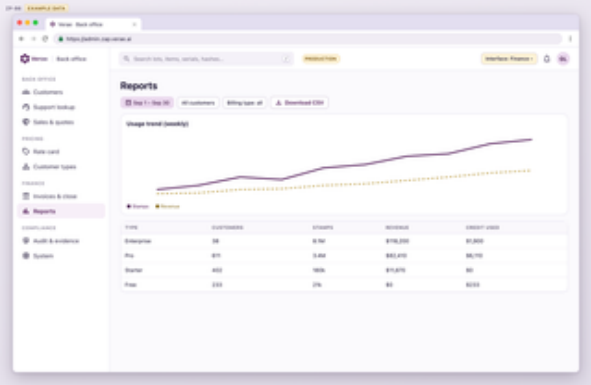
API id	Method	Path	Key params (name:type)
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-B6 — Finance — reports

Date-ranged billing reports by customer/type with CSV and usage-trend charts (admin Reports tab).

web



ZP-B6

Features	
• Filters + CSV.	• Trend chart (stamps vs revenue).
• Breakdown by type.	

Function / source references

`veb/ (verae-edge-billing) admin data-tab=reports`

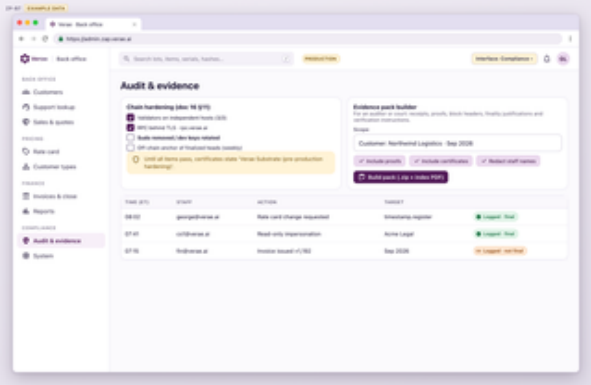
API id	Method	Path	Key params (name:type)
`portal.billing`	**GET/POST**	`customer.verae.ai/portal/api/billing/*`	methods:GET, reload:POST {amountCents:number}, po-request:POST {poNumber,terms}
`portal.profile`	**GET/PATCH**	`customer.verae.ai/portal/api/profile`	Authorization:header string, name,companyName,groupName,…:string, billingType:'stripe'|'purchase_order', phone:string E.164
`portal.usage`	**GET**	`customer.verae.ai/portal/api/usage`	peergosUsername:string?, from,to:ISO date?
`portal.tokens`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/tokens`	name:string, id:string

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-B7 — Compliance — audit log, evidence packs, chain hardening

Staff audit log (itself stamped), evidence-pack export, and the chain-hardening checklist that gates production claims.

web



ZP-B7

Features	
• Hardening checklist gating certificate wording.	• Evidence pack builder.
• Staff audit log with receipt states.	

Function / source references
`verae-staff-iam Audit tab`
`itad docs 16 §11`
`proposal`

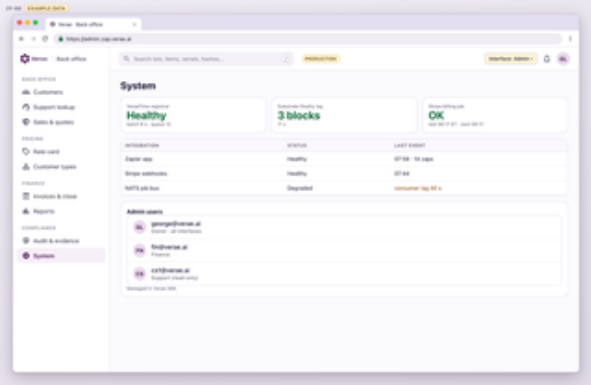
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-B8 — Admin — system health & users

Integration health (registrar, finality, Stripe job, Zapier, NATS) and admin users (admin System + Users tabs).

web



ZP-B8

Features	
• Health KPIs.	• Integration table.
• Admin users → IAM.	

Function / source references
`veb/ (verae-edge-billing) admin data-tab=system, users`
`NATS.md`

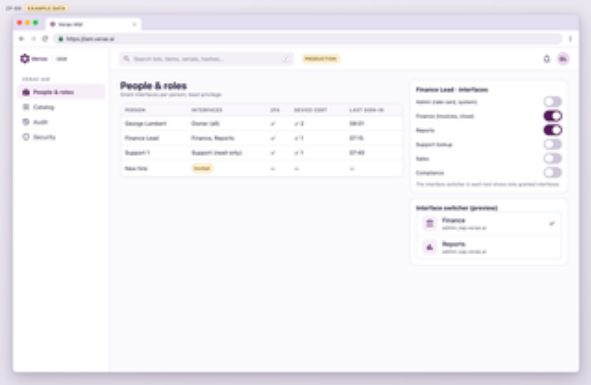
API id	Method	Path	Key params (name:type)
`peergos.cube`	—	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-B9 — Staff IAM — people, roles & interface switcher

Grant per-person interfaces (Admin/Finance/Support/Sales/Compliance), see 2FA/device-cert status; drives the interface switcher (verae-staff-iam).

web



ZP-B9

Features	
• People table with security posture.	• Per-interface toggles.
• Switcher preview.	

Function / source references
`verae-staff-iam/public (People & roles, Catalog, Audit, Security)`

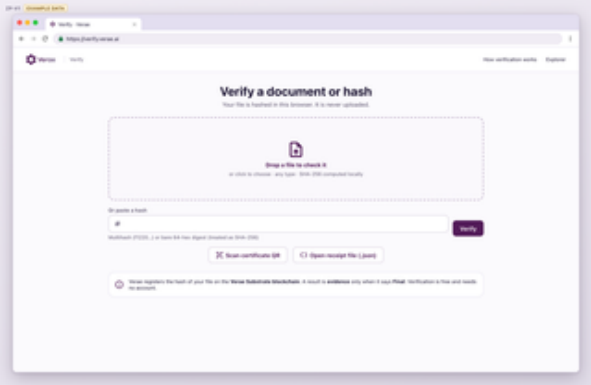
API id	Method	Path	Key params (name:type)
`portal.login`	**POST**	`customer.verae.ai/portal/api/login`	email:string, password:string, totpCode:string (6 digits)
`portal.2fa`	**GET/POST/DELETE**	`customer.verae.ai/portal/api/2fa/*`	setup:POST {}, code:string, totpCode:string
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-V1 — Verify — entry (file, hash, QR)

Public, account-free verification: drop a file (hashed locally), paste a multihash/hex, scan a certificate QR or open a receipt file.

web



ZP-V1

Features	
• Local hashing drop zone (Web Crypto; file never uploaded).	• Hash input: multihash or bare 64-hex = SHA-256.
• QR scan / receipt-file entry.	• Plain-language evidence rule.

Function / source references
`proposal`
`printit pdfcert.go (QR → explorer.verae.ai/time)`
`itad docs 09 §4`

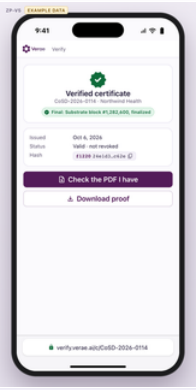
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-V2 — Verify — mobile (QR scan landing)

Landing page after scanning a certificate QR on a phone; can also check the PDF the user has.

web



ZP-V5

Features	
• Verdict card.	• Key facts.
• Compare with local copy.	

Function / source references

`proposal`

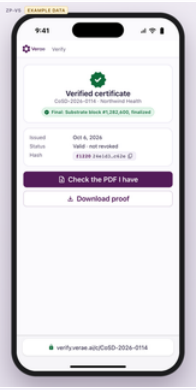
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-V3 — Verify — mobile (QR scan landing)

Landing page after scanning a certificate QR on a phone; can also check the PDF the user has.

web



ZP-V5

Features	
• Verdict card.	• Key facts.
• Compare with local copy.	

Function / source references

`proposal`

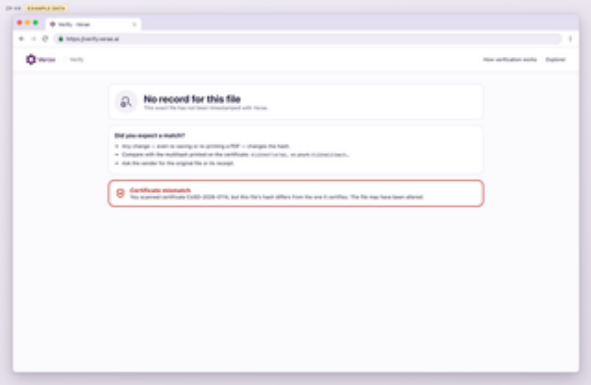
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-V4 — Verify — not found / tampered / revoked

Negative outcomes explained without blame; certificate mismatch is called out as possible tampering.

web



ZP-V4

Features	
• Neutral not-found verdict.	• Help: hash sensitivity, compare with the printed multihash.
• Mismatch vs scanned certificate (red). Revoked certificates show reason + successor.	

Function / source references
`proposal`
`printit pdfcert.go (QR → explorer.verae.ai/time)`
`itad docs 09 §4`

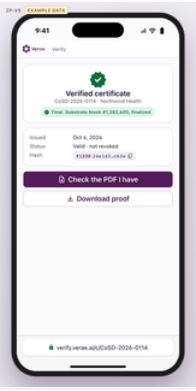
API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf

ZP-V5 — Verify — mobile (QR scan landing)

Landing page after scanning a certificate QR on a phone; can also check the PDF the user has.

web



ZP-V5

Features	
• Verdict card.	• Key facts.
• Compare with local copy.	

Function / source references

`proposal`

API id	Method	Path	Key params (name:type)
`v1.timestamp.status`	**GET**	`customer.verae.ai/v1/timestamp/status`	sha256:string (64 hex), jobId:string?
`peergos.cube`	**—**	`peergos.verae.ai (Peergos HTTP/API + local rust)`	username:string, password:string, path:string, bytes:bytes
`v1.timestamp`	**POST**	`customer.verae.ai/v1/timestamp`	sha256:string (64 hex), restamp:boolean, restampOf:string jobId?, metadata:object?

Sample calls + full param tables → VERAE-API-REFERENCE.pdf