

Volume VI

KLT/RBD: Software Layer, Database, Site and Legal- Technical Contour

English Edition · v8.2

Ivan Borisovich Kurpishev / Иван Борисович Курпишев
Independent Researcher · Kaliningrad · me@kurpishev.ru
2026

KLT-DOCTRINE-FINAL-MONOGRAPH-VOLUME-VI-KLT-RBD-SOFTWARE-LEGAL-EN-v8.2

Abstract

Volume VI assembles the software and registration contour of the Kurpishev Doctrine: KLT 4.14, KLT 5.1, RBD/RPD, real-data import, site publication, public/private split, deposit materials, and a cautious software-database-invention boundary. It is a technical and editorial binder, not a substitute for live legal filing review.

Annotation in Russian

Английская редакция Тома VI фиксирует программно-регистрационный контур Доктрины Курпишева: KLT 4.14, KLT 5.1, RBD/RPD, импорт реальных данных, сайт, public/private split, депозитные материалы и границу программа - база данных - изобретательский кандидат.

Notation dictionary

- C@C - event@state.
- Rep(R,I,U;D) - Reper quadruple.
- lambda - harmonic consistency measure.
- CGI - causal-graph gap index.
- RBD/RPD - Reper Database / Reper-Projective Database.
- Evidence-D - sufficient foundation suitable for audit.
- public/private split - separation of publishable and controlled materials.

0. Editorial status and v8.2 control point

Volume VI EN is the English counterpart of the Russian v8.1 technical-registration binder. It assembles the applied software, database, publication and legal-technical contour of the Kurpishev Doctrine: KLT 4.14, KLT 5.1, RBD/RPD, real-data import, site publication, deposit-material mapping and the cautious boundary between software, database and invention analysis.

Control point:

KLT-DOCTRINE-FINAL-MONOGRAPH-VOLUME-VI-KLT-RBD-SOFTWARE-LEGAL-EN-v8.2

This volume is not a final legal filing and does not replace live official-form verification. Its function is to create a reproducible technical corpus: source map, deposition map, boundary matrix, QA log, render verification and transport package.

1. Cross-volume formula of the software-registration layer

The governing chain of Volume VI is:

`Object -> C@C_x -> Rep_x(R,I,U;D) -> lambda_x -> CGI_x -> Status_x -> RBD_x -> publication/deposit route`.

A file, source archive, page, database row, report, build, checklist or legal draft is first fixed as event@state. It then receives a Reper quadruple: R is the factual content and executable state; I is the construction idea; U is the field of permitted uses; D is the sufficient foundation: source code, specification, checksum, test log, rendered page or official review note.

No software or legal-technical claim receives truth-status without Dom and D. Missing domain, missing verification environment, absent checksum, unclear version, private-data uncertainty or broken source binding produces a gap-status rather than an authorized result.

2. KLT 4.14 as checked demonstration and audit layer

KLT 4.14 is fixed as a checked transitional software build. It preserves the audit line of KLT 4.13 and expands it toward universal import, archive handling, web shell, mobile shell, report templates and self-test logistics.

Technical Reper:

```
`Rep_KLT414 = (R_build, I_lambda_check, U_import_export, D_verification)`.
```

The construction is interpreted as follows:

- R_build: Python package, CLI, web shell, DOCX templates, examples and self-test output;
- I_lambda_check: lambda-checklist, comparison against an ideal answer and ведомость-style reporting;
- U_import_export: TXT, DOC, DOCX, HTML, PDF, images, TEX, ODT, EPUB, ZIP and mass-output channels;
- D_verification: selftest.log, CHECK_RESULT, templates, renders, demos and checksum-bound package files.

KLT 4.14 is a concrete programmatic realization of applied Reper audit. It does not assert ownership over classical mathematical objects; it fixes the author's software architecture for applying KLT/RBD audit to documents, reports and mixed digital artifacts.

Table 1. KLT 4.14 - software layer

Component	Function	D-foundation
CLI Python engine	local audit and comparison with ideal answer	README, app.py, tests, selftest
Universal import	TXT/DOCX/HTML/PDF/images/TEX/ODT/EPUB/archives	importers.py, archives.py
DOCX reports	lambda checklists, score sheets, plagiarism sheet	templates, reports.py
Web shell	drag-and-drop interface	web/index.html, server.py
Flutter shell	mobile Android/iOS shell	mobile_flutter_shell

3. KLT 5.1 as SDK and project-graph bridge

KLT 5.1 is fixed as an SDK layer connecting the KLT 5.0 line, construction graph engine, PEAKS/PIX model, lambda defect, project finance indicators and visual outputs inside a cross-platform shell.

SDK formula:

```
`KLT5.1 = Flutter_UI + Python_reference_core + Project_JSON + lambda_engine + SVG/JSON/MD reports + verification`.
```

Computational center:

```
`lambda = ((U - R)(I - D)) / ((U - D)(I - R))`,
`delta = |lambda + 1|`,
`Auth = 1/(1 + delta)`.
```

The audited object is not merely a text but a project configuration: work items, resources, deliveries, estimates, contractual parameters, financial deviations, technological overlays and risk graphs. KLT 5.1 therefore occupies the bridge between the mathematical Reper core and the applied digital product.

Table 2. KLT 5.1 - SDK architecture

Component	Function	D-foundation
Flutter app	trilingual interface and visual tabs	app/lib, pubspec.yaml
Python reference core	verifiable calculation core	core_python/klt51_core.py, tests
Project JSON	works/resources/deliveries/costs/finance	examples/demo_project.json
Lambda engine	lambda, delta, Auth, D*, MSI, DI, CRI	docs/TZ, source code
SVG/MD/JSON outputs	schemes, charts, report, result	examples/out

4. RBD/RPD as database of Reper evidence and traceability graph

RBD/RPD is defined as a database of Reper objects, sources, versions, site pages, software builds, deposit materials and proof-status queues. It is not a flat bibliography or a folder index; it is a computable map of sufficient foundations.

Minimal schema:

```
`source -> artifact -> C@C -> Rep -> lambda_audit_item -> evidence_D -> status ->
release/deposit/publication link`.
```

After real-data import, artifacts are not stored only by file name. They receive roles: site_page, fips_package, publication_package, software_build, public_private_split, legal_review_event, lambda_audit_item and release_link. Publication and registration are therefore not performed directly from a directory; they pass through an RBD node, privacy status, legal-review queue and sufficient-foundation check.

Table 3. RBD/RPD - imported classes

Class	Meaning	Status
import_artifacts	real discovered artifacts	controlled register
site_pages	site pages and mirrors	public/controlled split
fips_packages	pre-filing package materials	live-check queue
software_builds	software packages and builds	sha/checksum binding
lambda_audit_items	lambda-audit nodes	gap/review mechanism
rbd_release_links	artifact-to-release mapping	rollback and release map

5. Real-data import and control queues

The package `KLT-RBD-SCHEMA-EXTENSION-REAL-DATA-IMPORT-v1.0` imports real project artifacts into a SQLite-based RBD/RPD layer. Website assets, PDF/HTML/MD/CSV/JSON/SQL/ZIP/checksum files, FIPS packages, software builds and publication packages become queryable and auditable objects.

The central discipline is negative: remaining queues are not treated as system failure. FIPS live-check queue, privacy review queue, lambda gaps and public/private split are part of the correct RBD architecture. They identify materials that must not be published, filed or represented as complete without additional foundation.

Traceability theorem. If every publishable or depositable object has source-id, artifact-id, Reper-id, checksum, privacy-status, legal-review-status and release-link, then the publication package reconstructs its origin chain without relying on informal project memory.

Figure 1. RBD/RPD real-data import flow.

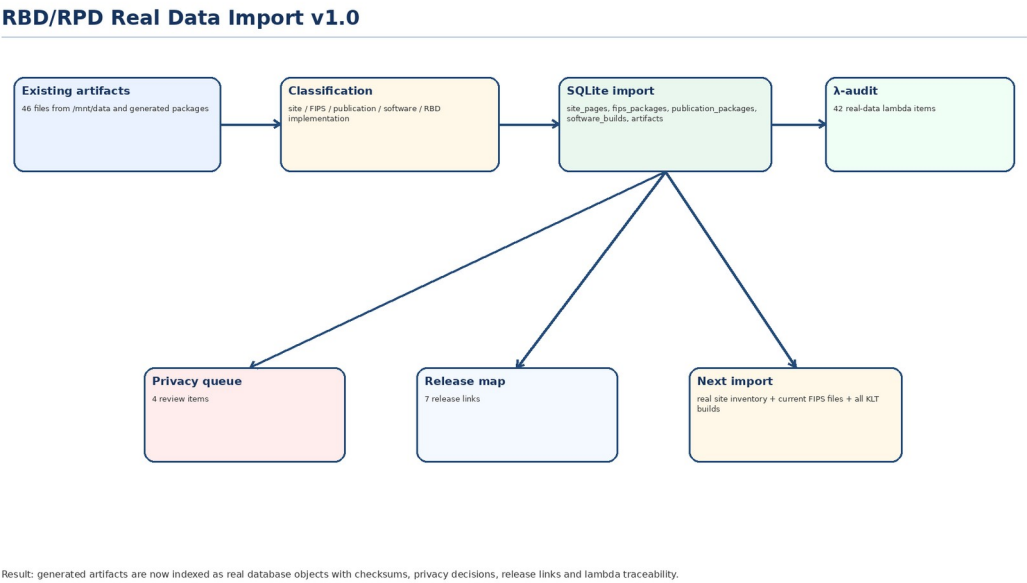
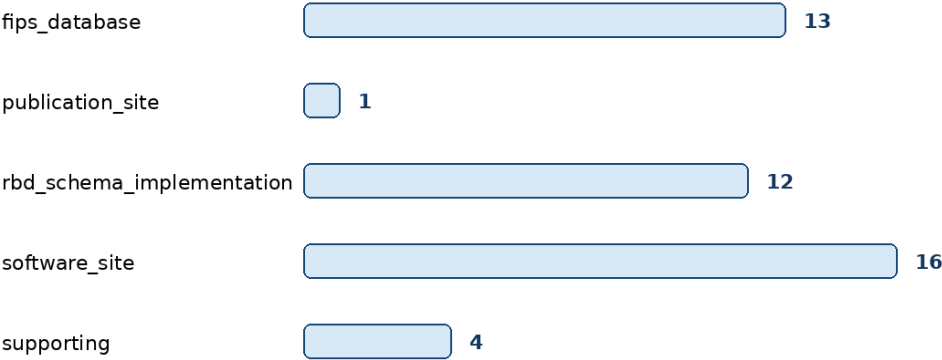


Figure 2. Imported artifacts by class.

Imported artifacts by class



6. The site as publication layer, not as truth source

The kurpishev.com site is treated as a public route, not as the primary truth source. The primary source remains the bundle: source package, manifest, RBD node, checksum, proof-status and QA.

Public route:

`source package -> site route -> public annotation -> download map -> checksum -> rollback point`.

For KLT 4.14/5.1 the short-route discipline is preserved: `/ru/klt/k414.html`, `/ru/klt/k51.html`, folders `a/`, `d/`, `s/`, `v/`, and maps such as `manifest.txt`, `map.json`, `nav.json` and `sitemap.xml`. Short names are not cosmetic; they make publication routes reproducible and portable.

7. Boundary between software, database and invention analysis

The legal-technical contour is divided into three cautious routes.

1. Software contour: KLT 4.14 and KLT 5.1 as concrete source-code, interface, reference-core, test, documentation, example and visualization packages.
2. Database contour: RBD/RPD as a structured system of tables, graph edges, records, imports, evidence-binding, release links and audit views.
3. Invention-analysis contour: only a candidate line for a computer-implemented method/system of processing heterogeneous digital documents and project data. This line does not assert pure formulas, abstract mathematics or a mathematical method as such. The candidate object, if developed further, is the technical architecture: input adapters, audit graph, gap localization, report generator and controlled reassembly.

The invention-analysis part of this volume is assigned the status `candidate / requires external patent review`, not the status of a completed patent filing.

Table 4. Legal-technical boundary

Layer	Route	Restriction
KLT 4.14 / KLT 5.1	software dossier	does not claim classical mathematics as property
RBD/RPD	database registration contour	personal and mixed artifacts require privacy review
KLT algorithmic method	computer-implemented method candidate	requires external patent review
Site layer	publication and public annotation	does not replace source of truth or official deposit

8. Deposit package and public/private split

Deposit material must be separated into public, controlled and restricted elements.

Public elements: site-ready pages, public annotations, brief README files, demo JSON/SVG/MD, general schemes and download routes.

Controlled elements: source ZIP packages, SQLite databases, CSV exports, SHA256 files, manifests, verification logs, technical descriptions, filing drafts and field maps.

Restricted or review-required elements: personal data, mixed artifacts, private documents, uncleaned requisites, intermediate legal-review notes and any material that may disclose excessive information before filing.

Rule: `public route` must not be created before `public/private split`. If the split is absent, the object is not published; it is moved to review.

9. QA contour of Volume VI EN

The QA contour contains five checks.

- Source QA: source files, roles and SHA256 hashes are listed.
- Document QA: DOCX, PDF, TEX and MD are created.
- Render QA: DOCX and PDF are rendered to PNG pages and contact sheets.
- Package QA: ZIP is created, SHA256 is generated and `unzip -t` is executed.
- Legal-technical QA: legal boundary matrix and deposit materials map are included; final official filing is reserved for a live procedural check.

Final v8.2 status: `ready as English monograph volume and technical-registration binder; not a final legal filing without live official-form verification`.

10. Definitions and theorems

Definition 10.1. A Software-Reper is a quadruple `Rep_SW = (R_code, I_method, U_runtime, D_verification)`, where D_verification includes source, checksum, test, documentation and runtime environment.

Definition 10.2. A Database-Reper is a quadruple `Rep_DB = (R_records, I_schema, U_queries, D_integrity)`, where D_integrity includes schema, export, checksum, SQL queries and integrity-check log.

Definition 10.3. A Deposit-Reper is a quadruple `Rep_DEP = (R_materials, I_claim_route, U_registration_routes, D_formal_basis)`, where D_formal_basis is not replaced by the intention to file; it requires current forms, material list, identity/author data and privacy review.

Theorem 10.4. If a software or database object has C@C fixation, a Reper quadruple, checksum, verification log and RBD binding, then it admits a reproducible technical audit trail.

Proof. C@C fixes event and state. Reper supplies the four mandatory components. SHA256 fixes file identity. The verification log connects the file to an execution or review environment. RBD binding places the object in a source-artifact-evidence-status graph. Hence the object can be reconstructed as `source -> artifact -> evidence -> status`.

Corollary 10.5. The absence of any of these components does not destroy the object; it moves the object from ready-for-deposit status to a gap queue.

Appendix A. Source binding

Table A1. Source binding

Source	Size	SHA256	Role
monograph5_0_ru.pdf	14300260	ea84bab141342eb6...	Master corpus 5.0: C@C, Rep, lambda, KPF/RPHD, KLT/RBD applications and

			source-preservation discipline.
KLT_4_14_CHECKED_BUILD.zip	1170911	5d48e7ce184527fb...	Checked KLT 4.14 build: universal import, web shell, mobile shell, checklist and report workflow.
KLT5_1_FLUTTER_SDK_PACKAGE.zip	3019939	d373e55bab5e63fa...	KLT 5.1 Flutter SDK: cross-platform application, Python reference core, examples and verification.
klt_rbd_schema_extension_real_data_import_v1_0_fixed.zip	11969199	406632a118589a45...	RBD/RPD schema extension and real-data import: SQLite, imported artifacts, site pages, FIPS queues and lambda audit items.
kurpishev_com_klt414_fixed_site_package.zip	4714585	9b334cdfa2bcdf2d...	Short-name website logistics for KLT 4.14/5.1.
kurpishev_site_responsive_polished_pack.zip	18759229	89713f360a21100a...	Responsive site publication layer with KLT pages and download assets.
KLT_DOCTRINE_MONOGRAPH_SOURCE_v7_0_PACKAGE (1).zip	35008439	b50a0408fa8f745d...	Doctrine monograph source skeleton and assembly protocol v7.0.
KLT_DOCTRINE_VOL6_RU_v8_1_PACKAGE.zip	1992535	991b34b205d78efb...	Russian Volume VI v8.1 source package used as translation and synchronization basis.

Appendix B. Control CSV files

Control files created: LEGAL_BOUNDARY_MATRIX_VOL6_EN_v8_2.csv and DEPOSIT_MATERIALS_MAP_VOL6_EN_v8_2.csv. They are included in the transport ZIP and serve as the legal-technical boundary index and deposit-material map.

Next point

KLT-DOCTRINE-FINAL-MONOGRAPH-MASTER-INDEX-SITE-BUNDLE-RU-EN-v8.3

Master index, cross-volume site-ready binding and transport map for the RU/EN doctrine volumes and application packages.