

CDS COMPOSITE DESIGN SOFTWARE

MASTER INFRASTRUCTURE AND BACKEND OPERATIONS MANUAL

Website • WB • Licensing • Engineering Data • Web API • Compiled MATLAB • Git/GitLab • Release
Operations

PROCESS BASELINE 2.0

Engineering baseline: Web r17 • Model ID 2026082901 • API CDS-WEB-1.0

29 August 2026

Document control

Document	CDS Master Infrastructure and Backend Operations Manual
Release	CDS Web r17 / web-r17
Model / API	2026082901 / CDS-WEB-1.0
Process baseline	2.0
Authority	Governed Git repository and release manifests
Publication targets	GitLab, CDSComposites.com, WB, backend deployment, archive
Release date	29 August 2026

This controlled manual updates the infrastructure, licensing, backend, user, release, publication, and recovery procedures for the browser-first CDS Web r17 platform. Native MATLAB Compiler SDK/Runtime R2026a acceptance passed; production activation remains governed by the target-specific deployment checkpoints stated in this manual.

Contents

Sections 1–9	Sections 10–18
1. Purpose	10. Runtime acceptance gate
2. Governed command lifecycle	11. Observability and evidence
3. Authority and identity	12. Failure and recovery
4. Platform actors	13. Release and GitLab synchronization
5. Web and cloud boundary	14. Publication and rollback
6. Licensing and authorization	15. Operational runbooks
7. Engineering records and Recipes	16. Infrastructure flowchart standard
8. Backend execution architecture	17. Release inventory
9. MATLAB package boundary	18. Handoff checklist

Four-command quick reference

Command	Controlled outcome
Model	Implement and validate a governed engineering capability.
Recipe	Freeze a dependency-complete reproducible configuration.
Release	Validate, package, commit, tag, and record an immutable identity.
Publish	Deploy that exact identity to approved user-facing targets.

CDS COMPLETE PLATFORM INFRASTRUCTURE

Web r17 • Model 2026082901 • API CDS-WEB-1.0 • Process baseline 2.0

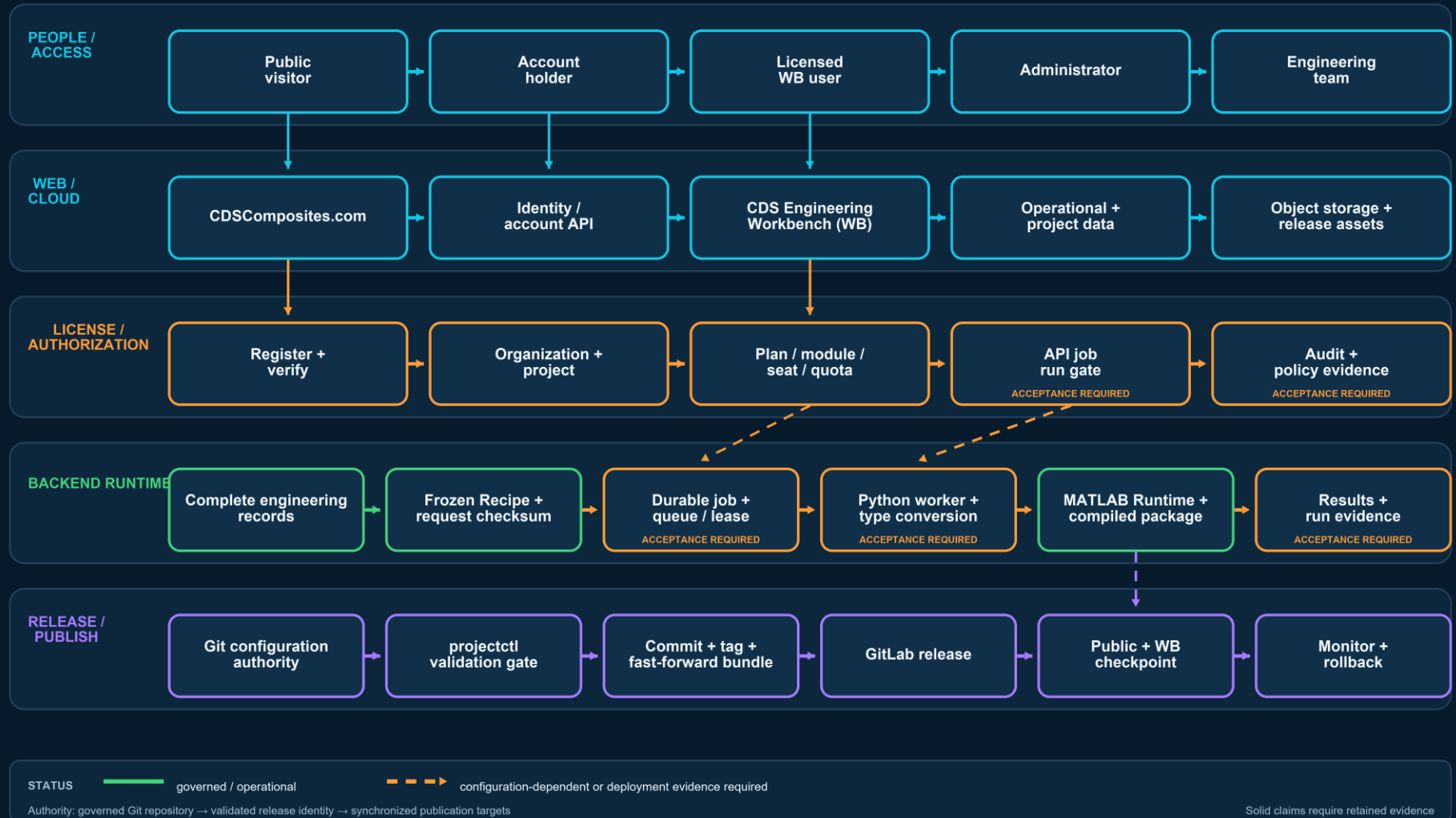


Figure 1. Complete Web r17 infrastructure baseline. Solid paths are governed or operational; dashed orange paths require configuration or target-specific deployment evidence.

01 Purpose

This manual governs the browser-first CDS platform from authenticated user access through engineering records, Recipes, queued solver execution, retained results, release control, publication, and recovery. The Git-controlled repository and its manifests remain the configuration authority. CDSComposites.com, CDS Engineering Workbench (WB), GitLab releases, download packages, and archives are synchronized publication targets.

The Web r17 release source is complete and statically validated. Native MATLAB R2026a, MATLAB Compiler SDK packaging, matching MATLAB Runtime execution, and controlled direct-versus-compiled comparison passed on 29 August 2026. Production activation still requires the target API, queue, worker, authorization, storage, monitoring, and rollback checkpoints described in this manual.

02 Governed command lifecycle

- Model implements or revises a governed engineering capability.
- Recipe freezes a dependency-complete, reproducible engineering configuration.
- Release validates, packages, commits, tags, and records an immutable identity.
- Publish deploys that exact identity to approved user-facing destinations.

03 Authority and identity

The release identity must agree across the MATLAB entry point, API guide, request example, database metadata, WB presentation, infrastructure map, manual, release notes, checksum evidence, Git tag, GitLab release, and public downloads.

Controlled item	Web r17 value
MATLAB entry point	CDS_USolve_Web_v2026_08_29_r17(web_request)
Request/response schema	CDS-WEB-1.0
Model ID	2026082901
Group lengths	[17,16,2,10,4,16,22,23]
Traceability results	O1-O68
Process-cycle steps	2-100
Database schema	1.1.0
Git tag	web-r17

04 Platform actors

- Public visitors browse capabilities, theory, examples, downloads, and contact routes.
- Account holders verify identity, maintain profiles, request access, and join organizations or projects.
- Licensed WB users create and edit complete engineering records, construct Recipes, submit authorized jobs, and review retained evidence.
- Administrators manage accounts, plans, module entitlements, organization access, quotas, audit events, and publication checkpoints.
- Engineering authors maintain the governed model, schema, validators, manuals, examples, and release evidence.

No browser control alone is trusted to enforce licensing. Authorization is checked by the API before a job reaches the queue and again by the worker before the compiled package runs.

05 Web and cloud boundary

CDSComposites.com owns public documentation, registration, account, support, licensing, and download experiences. WB owns engineering record editing, dependency navigation, Recipe construction, process-cycle entry, job submission, and results presentation.

Operational stores are separated by responsibility:

- Identity and operational data: accounts, sessions, organization membership, entitlements, license events, jobs, quotas, and audit events.
- Engineering project data: complete material, lamina, laminate, process, geometry, load, analysis, Recipe, and run records.
- Queue and leases: job order, worker ownership, progress, cancellation, and retry state.
- Object storage: immutable input snapshots, large result arrays, plots, reports, logs, and release artifacts.
- Static publication: public pages, theory, help, route maps, manifests, and downloads.

06 Licensing and authorization

The browser-first sequence is:

1. The user registers and verifies the CDS account.
2. Organization/project membership is established.
3. An administrator or approved commercial rule assigns plan, module, seat, quota, and validity constraints.
4. The identity service issues a short-lived authenticated session.
5. WB submits a Recipe job request.
6. The API checks account status, organization membership, module entitlement, compatible model ID/API schema, Recipe ownership, and quota.
7. The worker repeats the authorization check before invoking MATLAB Runtime.
8. The run record retains the authorization decision, immutable request checksum, model ID, Recipe ID, status, results, and audit linkage.

Signing keys, session secrets, database credentials, queue credentials, object-storage credentials, email/payment secrets, and Git credentials never enter public assets, WB client code, solver source, examples, release ZIPs, or Recipes.

07 Engineering records and Recipes

WB stores complete property records, including values that are not active in the current Recipe. Properties consumed by the active Recipe are highlighted. References use compatible selection controls and direct navigation; they are never entered as free-text identifiers. Deletion requires a dependency warning whenever downstream records would be disconnected.

Materials, laminae, laminates, process cases, structures, loads, analyses, and Recipes use stable IDs, canonical SI values, explicit units, property-level sources, foreign keys, transactions, and schema versions. NULL means unknown and is never silently converted to zero.

Process cycles accept 2-100 ordered steps containing time, temperature, pressure, and time-varying upper and lower boundary conditions. Pressure is stored and transported by Web r17 but is not yet coupled into the governing mechanical equations; responses must expose that limitation.

A runnable Recipe contains:

- Recipe identity, owner, status, compatible release, and schema.
- Dependency closure for every referenced engineering record.
- Canonical semantic inputs plus the exact resolved web request.
- Procedure and expected result summaries.
- Validation evidence and known limitations.
- File/record inventory with SHA-256 values.

SQLite .cdsdb remains the portable engineering interchange and recovery format. Multi-user operational storage may use a server database while preserving the same semantic entities and dependency rules.

08 Backend execution architecture

The production target path is:

```
WB -> authenticated API -> authorization/validation -> durable job -> queue -> Python worker ->
MATLAB Runtime -> compiled CDS package -> result/evidence storage -> WB
```

API responsibilities

- Authenticate user and organization context.
- Verify entitlement, module, ownership, compatibility, and quota.
- Freeze the Recipe and create an immutable request snapshot.
- Query project data and resolve dependency closure.
- Validate the CDS-WEB-1.0 request before queueing.
- Expose health, contract, submit, status, results, cancellation, and artifact endpoints.
- Serialize results and return stable job/status identifiers.

Worker responsibilities

- Claim one leased job and repeat the run authorization check.
- Initialize MATLAB Runtime and the compiled package once per worker process.
- Convert JSON-safe values to MATLAB numeric arrays and structures.
- Run one solver job at a time per package handle.
- Convert web_response to JSON-safe named families and O1-O68.
- Persist immutable inputs, results, diagnostics, timing, checksums, and artifacts.
- Release or fail the lease cleanly and recycle unhealthy workers.

Job states

Queued -> Running -> Completed | Failed | Cancelled

Retries create explicit attempt records. A retry never overwrites a completed or failed attempt. Cancellation is cooperative: the API records intent, the queue/worker observes it, and the final state explains whether execution stopped before or after solver invocation.

09 MATLAB package boundary

CDS_USolve_Web_v2026_08_29_r17 accepts one web_request and returns one web_response. The compiled package owns the governed equations and numerical response. The Python service owns authentication, database queries, Recipe resolution, queuing, type conversion, serialization, evidence persistence, and HTTP behavior.

web_response.outputs contains O1-O68. Named result families include primary, laminate, micromechanics, transport, Double-Double, structural, discontinuous-fiber, and diagnostics results.

Headless execution keeps plotting, animation, file, report, and verbosity controls disabled by default. Large histories and artifacts are persisted by the service, not emitted through console output.

10 Runtime acceptance gate

Before production numerical execution:

1. Compile the Web r17 entry point with the approved MATLAB release and MATLAB Compiler SDK.
2. Install the exactly matching MATLAB Runtime release/update on the worker host.
3. Initialize the generated Python package once and retain the returned handle.
4. Execute controlled plate, cylinder, beam, process, and aligned-discontinuous Recipes.
5. Compare direct MATLAB, compiled package, worker, API, and WB results using named outputs and O1-O68.
6. Record tolerances, environment, package/runtime versions, timing, checksums, and reviewer approval.
7. Enable production workers only after every required comparison passes.

Static checks are necessary but are not numerical runtime acceptance.

11 Observability and evidence

Every job records:

- User, organization, project, Recipe, model ID, API schema, worker, and attempt.
- Authorization decision and applicable module/quota policy.
- Immutable request and dependency checksums.
- Queue, start, completion, and cancellation timestamps.
- Worker/package/runtime versions and health.
- Named/O1-O68 response checksums.
- Warnings, limitations, structured errors, logs, and artifact links.

Health is separated into API, database, queue, object storage, worker, MATLAB Runtime, compiled package, and end-to-end reference-Recipe checks. A healthy API does not prove a healthy numerical backend.

12 Failure and recovery

- Validation failures stop before queueing and return field-level diagnostics.
- Authorization failures create an audit event but never create a solver job.
- Queue/lease failures preserve the immutable job and may be retried under policy.
- Worker or runtime failures retain stdout/stderr, structured diagnostics, attempt state, and host/package identity.
- Partial numerical results are never presented as completed engineering results.
- Object-storage failures prevent completion until result evidence is durably persisted.
- Recovery begins from the Git/release identity, then databases and object storage, then API/queue/workers, then WB and public publication targets.

13 Release and GitLab synchronization

The Web r17 fast-forward bundle begins at the immutable Web r16 commit 978b7fe66f92092452ca738c6ef26dbe07cc8a2d. The local merge script must verify that base, require a clean worktree, fetch the bundled main/tag references, fast-forward main, run project validation, and push main plus web-r17 to the official GitLab remote.

The formal GitLab release must point to the exact web-r17 tag and use the same package/checksums exposed by the public CDS download page.

14 Publication and rollback

Publication targets are GitLab, CDSComposites.com, WB, backend deployment, and archive/backup. Each target retains a rollback reference. Failed verification stops distribution and restores the prior healthy checkpoint without rewriting Git history or release evidence.

15 Operational runbooks

New licensed WB user

- Verify the account and organization membership.
- Assign plan, modules, validity, seat, and quota.
- Confirm WB access and API authorization.
- Submit a non-production validation Recipe.
- Verify the run/audit record and results access.

Backend deployment

- Verify the web-r17 tag, release manifest, package checksum, and deployment environment.
- Install matching MATLAB Runtime and generated Python package.
- Apply database migrations and verify queue/object-storage connectivity.
- Start one worker, run health and reference-Recipe acceptance, then scale.
- Confirm authorization, cancellation, failure, retry, result, and audit paths.
- Retain prior deployment and database rollback checkpoints.

Periodic checks

- Daily: public site, authentication, API, queue depth, worker health, failures, and backups.
- Weekly: download hashes, reference-Recipe execution, failed notifications, audit review, and restore sample.
- Per release: full validation, runtime parity, manifest/hash, deployment, and archive checks.
- Quarterly: key/secret inventory, access review, provider/dependency review, and disaster-recovery exercise.

16 Infrastructure flowchart standard

The controlled baseline is a 36 x 24-inch landscape editable SVG with print-ready one-page PDF and 7200 x 4800 PNG exports. The lanes are People/Access, Web/Cloud, License/Authorization, Backend Runtime, and Release/Publish. Cyan represents access/web, orange authorization, green engineering runtime, purple release/publication, and dark

navy the system boundary. Solid paths are governed/operational. Dashed orange paths are configuration-dependent, not accepted, or awaiting target-specific deployment evidence.

17 Release inventory

- Web r17 MATLAB master and validator.
- Web API/compilation guide and structured request example.
- Backend infrastructure and user manual source, DOCX, and PDF.
- Editable infrastructure SVG, print PDF, and 7200 x 4800 PNG.
- Twenty-four schema-1.1.0 sample databases and manifest.
- 160-record WB baseline material library.
- Project/release manifests, checksums, release notes, validation evidence, Git bundle, and local merge/push script.

18 Handoff checklist

- ☐ Repository and release identity agree.
- ☐ Web-only request/response contract and database schema are current.
- ☐ Manual and infrastructure diagram match the actual backend status.
- ☐ Static/database/archive gates pass.
- ☐ Native MATLAB Compiler SDK runtime acceptance is explicitly recorded.
- ☐ Git bundle base and head are verified.
- ☐ Local merge/push script requires a clean worktree and fast-forward.
- ☐ GitLab tag/release and public downloads reference the same bytes.
- ☐ Rollback and recovery references are retained.

Approval record

Role	Name	Decision / date	Evidence reference
Engineering			
Backend operations			
Release management			