



Document	Maintenance Documentation	Release	17/08/26 V1.0
Project	BRN WMS	Project code	200-WMS-26-001-00
Project period	05/01/26–24/08/26		

Note: the example reference package filed the same content twice under Product Operation Guide and Maintenance Documentation. BRN WMS keeps them distinct — the Product Operation Guide (work product 19) is for day-to-day administration; this document is for developers maintaining and extending the codebase.

Objective

Give a developer maintaining BRN WMS enough architectural context, component ownership, and known-issue awareness to make a safe, correctly-scoped change.

1. Architecture summary

See Software Design (work product 12) for the full component/deployment diagrams. In summary: PHP presentation + business-logic manager classes → two MariaDB databases (wms identity/company, wms2 WMS/accounting), plus a Node.js real-time/scheduler tier reached only through a secret-protected internal endpoint.

2. Component ownership

See Software Components (work product 14) for the full inventory. When changing behavior, locate the owning manager class first (e.g., stock behavior → `StockManager/StockSourceManager/StockTablesTrait`; posting/GL behavior → `PostingManager`; document numbering → `DocumentNumberManager`) rather than editing page-level code directly, consistent with NFR-007 (maintainability).

3. Database change procedure

- The `schema_migrations` table exists in the WMS database, indicating an intended migration-tracking mechanism; confirm the current migration convention before hand-editing schema in a shared environment.
- `setup.php` is the authoritative one-shot schema definition for a fresh environment; any schema change should be reflected there so a new environment matches production.
- Prefer additive, backward-compatible schema changes; coordinate destructive schema changes through the Change Report.

4. Configuration points

Area	File(s)	Notes
Database/app config	<code>app/config.php</code> (generated)	Never hand-edit the committed template with live secrets
Docker build	<code>docker/php/config.php.template</code> , <code>docker/php/entrypoint.sh</code>	Change here to affect all container deployments
Node.js services	<code>nodejs/server.js</code> , <code>nodejs/scheduler.js</code> , <code>nodejs/ecosystem.config.js</code>	Scheduled-job timing and notification wiring
CORS/notification security	<code>app/config.php</code> (<code>NODE_EMIT_SECRET</code>), Node.js CORS whitelist	See Correction Register entries on CORS/notify guarding

5. Known issues and defect history

The Correction Register (work product 4) is the authoritative known-issue history: 29 recorded corrections, all currently "Implemented; verification pending." Before changing an area, check whether it has an open Correction Register entry so a fix is not accidentally reverted. Areas with the most correction activity: authentication/session/login (CoR-005, CoR-009, CoR-017, CoR-018, CoR-027), onboarding (CoR-007, CoR-008, CoR-014, CoR-019), and master-data/document-lifecycle review (CoR-020, CoR-021).

6. Release procedure

1. Implement and locally verify the change. 2. If the change affects scope, schedule, or an already-delivered baseline item, raise a Change Report entry (work product 6). 3. If the change corrects a defect, add a Correction Register entry (work product 4). 4. Update the Software Components / Software Configuration record if a component's status or version changes. 5. Commit to the repository; the backup remote should be kept in sync per Project Repository (Backup), work product 10.

7. Approval

Prepared by

Name: Thanakorn Sathitwitayakul

Role: Developer

Signature: _____

Date: _____

Reviewed by

Name: Apirach Supattaratpateep

Role: Project Manager

Signature: _____

Date: _____

Reviewed and authorized by

Name: Seri Viriyasakultorn

Project roles: Project Sponsor / Customer Representative / Authorized Approver

Position: Managing Director

Company: B.R.N. Enterprise Co., Ltd.

Signature: _____

Date: _____