How it Works

Project Forge WMS technical and operational architecture

1. Purpose and Deployment Model

System Update installs an approved Project Forge WMS GitHub Release on a staging or production server. It downloads a packaged, checksummed release. It does not run `git pull`, require a Git working tree, copy staging files to production, or invoke Plesk Git deployment.

Developer → Git tag → GitHub Release → Staging → Production

Staging and production independently download the same immutable release assets from GitHub.

2. Major Components

Component	Responsibility
`httpdocs/manifest.json`	Application version, numeric build, requirements, database version, release date, and release notes.
`.github/workflows/project-forge-release.yml`	Validates the tag, builds the ZIP, calculates SHA-256, and creates a GitHub prerelease.
`httpdocs/modules/system_update/`	Administrator UI, AJAX endpoints, release discovery, validation, installation, backup, and status.
`ScheduleScripts/system_update.php`	CLI worker that claims and executes one queued update job.
`system_jobs`	Stores queued, running, completed, and failed jobs.
`APP_ROOT/storage/system_updates/`	Private downloads, extraction, backups, locks, logs, and installed-file inventory.
`APP_ROOT/config/system_update.php`	Private channel and GitHub token shared by web PHP and the CLI worker.

3. Release Channels and Immutability

Channel	Eligible releases	Use
`staging`	Non-draft prereleases and stable releases	Test before production approval.
`production`	Non-draft stable releases only	Install explicitly promoted releases.

Every new tag is published as a prerelease. After staging installs and approves it, an operator edits the same GitHub Release and clears its prerelease flag. Promotion must not create another tag, rebuild the ZIP, replace an asset, or change the checksum.

4. Check for Updates

1. The browser calls `modules/system_update/ajax/check_update.php`.
2. The endpoint validates the local manifest.
3. The GitHub Releases API is queried using the private token.
4. Drafts and releases disallowed by the configured channel are removed.
5. The updater requires an exactly named ZIP and SHA-256 asset.
6. The remote manifest is validated and compared with the local manifest.
7. Update is enabled only when a valid newer release is identified.

A release is newer when its numeric build is greater, or when builds tie and its semantic version is greater.

Read-only: Check for Updates does not modify files or queue installation.

5. Queue, Monitor, and Execute

1. `ajax/start_update.php` validates POST, authorization, CSRF, tag, channel, and concurrency.
2. It creates a `system_update` job with status `queue`.
3. The browser polls `job_status.php`. Polling only reads status; it does not execute the job.
4. Plesk runs `ScheduleScripts/system_update.php`.
5. The worker atomically claims the job and executes the update service.

Plesk: use Scheduled Task type *Run a PHP script*, not *Run a command*. Command tasks are chrooted and may not contain PHP.

```
Script: ScheduleScripts/system_update.php
PHP: 8.5
Schedule: * * * * *
```

6. Installation Phases

Phase	Operation
download	Resolve metadata and download ZIP/checksum using authenticated GitHub asset API URLs.
verify	Compare ZIP SHA-256 with the published checksum.
extract	Reject traversal, absolute paths, unsafe links, and size/count violations; extract privately.
preflight	Validate runtime, protected paths, storage placement, package layout, disk space, and migrations.
backup	Back up managed overwritten/removed files and record new files.
install	Overlay managed files and remove only obsolete files proven by a trusted inventory.
migrate	Run pending SQL migrations when present.
post-update	Verify required files and installed manifest; write new inventory.
cleanup	Remove temporary data and release the lock.
complete / failed	Record terminal state and administrator message.

7. Security Controls

- Token and channel are stored outside `httpdocs` and never returned to the browser.
- Updater storage must resolve outside `DOCUMENT_ROOT`.
- Downloads require HTTPS, approved GitHub hosts, authenticated asset API URLs, and approved redirect hosts.
- SHA-256 is verified before extraction.
- CSRF is mandatory for the update POST.
- Atomic job claiming and a lock prevent concurrent installation.
- Configuration, uploads, attachments, environment files, storage, logs, and backups are protected.

The private configuration file returns this array:

```
return [
    'channel' => 'staging', // production on production
    'github_token' => 'TOKEN_VALUE',
];
```

Recommended private-config mode: `0640`, owned by the website user and subscription group.

8. Storage and Logs

```
APP_ROOT/config/system_update.php
APP_ROOT/storage/system_updates/downloads/
APP_ROOT/storage/system_updates/extracted/
APP_ROOT/storage/system_updates/backups/
APP_ROOT/storage/system_updates/locks/
APP_ROOT/storage/system_updates/logs/
APP_ROOT/ScheduleScripts/system_update.php
```

Job logs use `storage/system_updates/logs/{job_id}.log`.

9. Backup and Recovery Limits

Before migrations start, rollback attempts to restore overwritten files, remove newly created managed files, and restore obsolete managed files removed during installation.

Database: updater file backups are not database dumps. SQL migrations are not automatically rolled back and may partially apply. A verified external database backup and recovery plan are required before migrations.

10. Definition of Success

- Job status is `done` and phase is `complete`.
- Local manifest equals the installed release version and build.
- A new check no longer offers the installed release.
- Protected paths and private configuration remain unchanged.
- Critical WMS smoke tests pass.
- The installed artifact checksum equals the approved GitHub checksum.

Created 2026-07-21 18:09:33 UTC by Andrew
Updated 2026-07-21 18:09:43 UTC by Andrew