

# changelog.md for @push.rocks/smartregistry

## 2026-03-27 - 2.8.2 - fix(maven,tests)

handle Maven Basic auth and accept deploy-plugin metadata/checksum uploads while stabilizing npm CLI test cleanup

- Validate Maven tokens from Basic auth credentials by extracting the password portion before token validation.
- Return successful responses for PUT requests to checksum and maven-metadata endpoints so Maven deploy uploads do not fail when files are auto-generated.
- Improve npm CLI integration test isolation and cleanup by using a temporary test directory, copying per-package .npmrc files, and cleaning stale published packages before test runs.
- Tighten test teardown by destroying the registry explicitly and simplifying package/install fixture generation.

## 2026-03-24 - 2.8.1 - fix(registry)

align OCI and RubyGems API behavior and improve npm search result ordering

- handle OCI version checks on /v2 and /v2/ endpoints
- return RubyGems versions JSON in the expected flat array format and update unyank coverage to use the HTTP endpoint
- prioritize exact and prefix matches in npm search results
- update documentation to reflect full upstream proxy support

# 2026-03-24 - 2.8.0 -

## feat(core,storage,oci,registry-config)

add streaming response support and configurable registry URLs across protocols

- Normalize SmartRegistry responses to ReadableStream bodies at the public API boundary and add stream helper utilities for buffers, JSON, and hashing
- Add streaming storage accessors for OCI, npm, Maven, Cargo, Composer, PyPI, and RubyGems downloads to reduce in-memory buffering
- Make per-protocol registryUrl configurable so CLI and integration tests can use correct host and port values
- Refactor OCI blob uploads to persist chunks in storage during upload and clean up temporary chunk objects after completion or expiry
- Update tests and storage integration to use the new stream-based response model and smartstorage backend

# 2025-12-03 - 2.7.0 -

## feat(upstream)

Add dynamic per-request upstream provider and integrate into registries

- Introduce IUpstreamProvider and IUpstreamResolutionContext to resolve upstream configs per request.
- Add StaticUpstreamProvider implementation for simple static upstream configurations.
- Propagate dynamic upstream provider through SmartRegistry and wire into protocol handlers (npm, oci, maven, cargo, composer, pypi, rubygems).
- Replace persistent per-protocol upstream instances with per-request resolution: registries now call provider.resolveUpstreamConfig(...) and instantiate protocol-specific Upstream when needed.
- Add IRequestActor to core interfaces and pass actor context (userId, ip, userAgent, etc.) to upstream resolution and storage/auth hooks.
- Update many protocol registries to accept an upstreamProvider instead of IProtocolUpstreamConfig and to attempt upstream fetches only when provider returns enabled config.

- Add utilities and tests: test helpers to create registries with upstream provider, a tracking upstream provider helper, StaticUpstreamProvider tests and extensive upstream/provider integration tests.
- Improve upstream interfaces and cache/fetch contexts (IUpstreamFetchContext includes actor) and add StaticUpstreamProvider class to upstream module.

## 2025-11-27 - 2.6.0 - feat(core)

Add core registry infrastructure: storage, auth, upstream cache, and protocol handlers

- Introduce RegistryStorage: unified storage abstraction with hook support (before/after put/delete/get) and helpers for OCI, NPM, Maven, Cargo, Composer, PyPI, and RubyGems paths and operations
- Add DefaultAuthProvider and AuthManager: in-memory token store, UUID tokens for package protocols, OCI JWT creation/validation, token lifecycle (create/validate/revoke) and authorization checking
- Add SmartRegistry orchestrator to initialize and route requests to protocol handlers (OCI, NPM, Maven, Cargo, Composer, PyPI, RubyGems)
- Implement upstream subsystem: UpstreamCache (in-memory + optional S3 persistence), BaseUpstream with multi-upstream routing, scope rules, retries, TTLs, stale-while-revalidate and negative caching
- Add circuit breaker implementation for upstream resilience with exponential backoff and per-upstream breakers
- Add protocol implementations and helpers: NpmRegistry/NpmUpstream (packument/tarball handling and tarball URL rewriting), PypiRegistry (PEP 503/691 support, uploads, metadata), MavenRegistry (artifact/metadata handling and checksum generation), CargoRegistry (sparse index, publish/download/yank)
- Utility exports and helpers: buffer helpers, plugins aggregator, path helpers, and various protocol-specific helper modules

## 2025-11-27 - 2.5.0 - feat(pypi,rubygems)

Add PyPI and RubyGems protocol implementations, upstream caching, and auth/storage improvements

- Implemented full PyPI support (PEP 503 Simple API HTML, PEP 691 JSON API, legacy upload handling, name normalization, hash verification, content negotiation, package/file storage and metadata management).

- Implemented RubyGems support (compact index, /versions, /info, /names endpoints, gem upload, yank/unyank, platform handling and file storage).
- Expanded RegistryStorage with protocol-specific helpers for OCI, NPM, Maven, Cargo, Composer, PyPI, and RubyGems (get/put/delete/list helpers, metadata handling, context-aware hooks).
- Added AuthManager and DefaultAuthProvider improvements: unified token creation/validation for multiple protocols (npm, oci, maven, composer, cargo, pypi, rubygems) and OCI JWT support.
- Added upstream infrastructure: BaseUpstream, UpstreamCache (S3-backed optional, stale-while-revalidate, negative caching), circuit breaker with retries/backoff and resilience defaults.
- Added various protocol registries (NPM, Maven, Cargo, OCI, PyPI) with request routing, permission checks, and optional upstream proxying/caching.

## 2025-11-27 - 2.4.0 - feat(core)

Add pluggable auth providers, storage hooks, multi-upstream cache awareness, and PyPI/RubyGems protocol implementations

- Introduce pluggable authentication: IAuthProvider interface and DefaultAuthProvider (in-memory) with OCI JWT support and UUID tokens.
- AuthManager now accepts a custom provider and delegates all auth operations (authenticate, validateToken, create/revoke tokens, authorize, listUserTokens).
- Add storage hooks (IStorageHooks) and hook contexts: beforePut/afterPut/afterGet/beforeDelete/afterDelete. RegistryStorage now supports hooks, context management (setContext/withContext) and invokes hooks around operations.
- RegistryStorage expanded with many protocol-specific helper methods (OCI, NPM, Maven, Cargo, Composer, PyPI, RubyGems) and improved S3/SmartBucket integration.
- Upstream improvements: BaseUpstream and UpstreamCache became multi-upstream aware (cache keys now include upstream URL), cache operations are async and support negative caching, stale-while-revalidate, ETag/metadata persistence, and S3-backed storage layer.
- Circuit breaker, retry, resilience and scope-rule routing enhancements for upstreams; upstream fetch logic updated to prefer primary upstream for cache keys and background revalidation behavior.
- SmartRegistry API extended to accept custom authProvider and storageHooks, and now wires RegistryStorage and AuthManager with those options. Core exports updated to expose auth and storage interfaces and DefaultAuthProvider.
- Add full PyPI (PEP 503/691, upload API) and RubyGems (Compact Index, API v1, uploads/yank/unyank, specs endpoints) registry implementations with parsing, upload/download, metadata management and upstream proxying.

- Add utility helpers: binary buffer helpers (toBuffer/isBinaryData), pypi and rubygems helper modules, and numerous protocol-specific helpers and tests referenced in readme.hints.
- These changes are additive and designed to be backward compatible; bumping minor version.

## 2025-11-27 - 2.3.0 - feat(upstream)

Add upstream proxy/cache subsystem and integrate per-protocol upstreams

- Introduce a complete upstream subsystem (BaseUpstream, UpstreamCache, CircuitBreaker) with caching, negative-cache, stale-while-revalidate, retries, exponential backoff and per-upstream circuit breakers.
- Add upstream interfaces and defaults (ts/upstream/interfaces.upstream.ts) and export upstream utilities from ts/upstream/index.ts and root ts/index.ts.
- Implement protocol-specific upstream clients for npm, pypi, maven, composer, cargo and rubygems (classes.\*upstream.ts) to fetch metadata and artifacts from configured upstream registries.
- Integrate upstream usage into registries: registries now accept an upstream config, attempt to fetch missing metadata/artifacts from upstreams, cache results locally, and expose destroy() to stop upstream resources.
- Add SmartRequest and minimatch to dependencies and expose smartrequest/minimatch via ts/plugins.ts for HTTP requests and glob-based scope matching.
- Update package.json to add @push.rocks/smartrequest and minimatch dependencies.
- Various registry implementations updated to utilize upstreams (npm, pypi, maven, composer, cargo, rubygems, oci) including URL rewrites and caching behavior.

## 2025-11-27 - 2.2.3 - fix(tests)

Use unique test run IDs and add S3 cleanup in test helpers to avoid cross-run conflicts

- Add generateTestRunId() helper in test/helpers/registry.ts to produce unique IDs for each test run
- Update PyPI and Composer native CLI tests to use generated testPackageName / unauth-pkg- to avoid package name collisions between runs
- Import smartbucket and add S3 bucket cleanup logic in test helpers to remove leftover objects between test runs

- Improve test robustness by skipping upload-dependent checks when tools (twine/composer) are not available and logging outputs for debugging

## 2025-11-25 - 2.2.2 - fix(npm)

Replace console logging with structured Smartlog in NPM registry and silence RubyGems helper error logging

- Replaced console.log calls with this.logger.log (Smartlog) in ts/npm/classes/npmregistry.ts for debug/info/success events
- Converted console.error in NpmRegistry.handleSearch to structured logger.log('error', ...) including the error message
- Removed console.error from ts/rubygems/helpers/rubygems.ts; gem metadata extraction failures are now handled silently by returning null

## 2025-11-25 - 2.2.1 - fix(core)

Normalize binary data handling across registries and add buffer helpers

- Add core/helpers.buffer.ts with isBinaryData and toBuffer utilities to consistently handle Buffer, Uint8Array, string and object inputs.
- Composer: accept Uint8Array uploads, convert to Buffer before ZIP extraction, SHA-1 calculation and storage.
- PyPI: accept multipart file content as Buffer or Uint8Array and normalize to Buffer before processing and storage.
- Maven: normalize artifact body input with toBuffer before validation and storage.
- OCI: improve upload id generation by using substring for correct random length.

## 2025-11-25 - 2.2.0 - feat(core/registrystorage)

Persist OCI manifest content-type in sidecar and normalize manifest body handling

- Add getOciManifestContentType(repository, digest) to read stored manifest Content-Type
- Store manifest Content-Type in a .type sidecar file when putOciManifest is called
- Update putOciManifest to persist both manifest data and its content type

- OciRegistry now retrieves stored content type (with fallback to detectManifestContentType) when serving manifests
- Add toBuffer helper in OciRegistry to consistently convert various request body forms to Buffer for digest calculation and uploads

## 2025-11-25 - 2.1.2 - fix(oci)

Prefer raw request body for content-addressable OCI operations and expose rawBody on request context

- Add rawBody?: Buffer to IRequestContext to allow callers to provide the exact raw request bytes for digest calculation (falls back to body if absent).
- OCI registry handlers now prefer context.rawBody over context.body for content-addressable operations (manifests, blobs, and blob uploads) to preserve exact bytes and ensure digest calculation matches client expectations.
- Upload flow updates: upload init, PATCH (upload chunk) and PUT (complete upload) now pass rawBody when available.

## 2025-11-25 - 2.1.1 - fix(oci)

Preserve raw manifest bytes for digest calculation and handle string/JSON manifest bodies in OCI registry

- Preserve the exact bytes of the manifest payload when computing the sha256 digest to comply with the OCI spec and avoid mismatches caused by re-serialization.
- Accept string request bodies (converted using UTF-8) and treat already-parsed JSON objects by re-serializing as a fallback.
- Keep existing content-type fallback logic while ensuring accurate digest calculation prior to storing manifests.

## 2025-11-25 - 2.1.0 - feat(oci)

Support configurable OCI token realm/service and centralize unauthorized responses

- SmartRegistry now forwards optional ociTokens (realm and service) from auth configuration to OciRegistry when OCI is enabled
- OciRegistry constructor accepts an optional ociTokens parameter and stores it for use in auth headers

- Replaced repeated construction of WWW-Authenticate headers with `createUnauthorizedResponse` and `createUnauthorizedHeadResponse` helpers that use configured `realm/service`
- Behavior is backwards-compatible: when `ociTokens` are not configured the registry falls back to the previous defaults (`realm: <basePath>/v2/token`, `service: "registry"`)

## 2025-11-25 - 2.0.0 - BREAKING CHANGE(pypi,rubygems)

Revise PyPI and RubyGems handling: normalize error payloads, fix `.gem` parsing/packing, adjust PyPI JSON API and tests, and export `smartarchive` plugin

- Rename error payload property from `'message'` to `'error'` in PyPI and RubyGems interfaces and responses; error responses are now returned as JSON objects (`body: { error: ... }`) instead of `Buffer(JSON.stringify(...))`.
- RubyGems: treat `.gem` files as plain tar archives (not gzipped). Use `metadata.gz` and `data.tar.gz` correctly, switch packing helper to pack plain tar, and use `zlib deflate` for `.rz` gemspec data.
- RubyGems registry: add legacy Marshal specs endpoint (`specs.4.8.gz`) and adjust versions handler invocation to accept request context.
- PyPI: adopt PEP 691 style (`files` is an array of file objects) in tests and metadata; include `requires_python` in test package metadata; update JSON API path matching to the package-level `'/{package}/json'` style used by the handler.
- Fix HTML escaping expectations in tests (`requires_python` values are HTML-escaped in attributes, e.g. `'>=3.8'`).
- Export `smartarchive` from plugins to enable archive helpers in core modules and helpers.
- Update tests and internal code to match the new error shape and API/format behaviour.

## 2025-11-25 - 1.9.0 - feat(auth)

Implement HMAC-SHA256 OCI JWTs; enhance PyPI & RubyGems uploads and normalize responses

- `AuthManager`: create and validate OCI JWTs signed with HMAC-SHA256 (`header.payload.signature`). Signature verification, `exp/nbf` checks and payload decoding implemented.
- PyPI: improved Simple API handling (PEP-691 JSON responses returned as objects), Simple HTML responses updated, upload handling enhanced to support nested/flat multipart fields, verify hashes (`sha256/md5/blake2b`), store files and return 201 on success.

- RubyGems: upload flow now attempts to extract gem metadata from the .gem binary when name/version are not provided, improved validation, and upload returns 201. Added extractGemMetadata helper.
- OCI: centralized 401 response creation (including proper WWW-Authenticate header) and HEAD behavior fixed to return no body per HTTP spec.
- SmartRegistry: use nullish coalescing for protocol basePath defaults to avoid falsy-value bugs when basePath is an empty string.
- Tests and helpers: test expectations adjusted (Content-Type startsWith check for HTML, PEP-691 projects is an array), test helper switched to smartarchive for packaging.
- Package.json: added devDependency @push.rocks/smartarchive and updated dev deps.
- Various response normalization: avoid unnecessary Buffer.from() for already-serialized objects/strings and standardize status codes for create/upload endpoints (201).

## 2025-11-24 - 1.8.0 - feat(smarts3)

Add local smarts3 testing support and documentation

- Added @push.rocks/smarts3 ^5.1.0 to devDependencies to enable a local S3-compatible test server.
- Updated README with a new "Testing with smarts3" section including a Quick Start example and integration test commands.
- Documented benefits and CI-friendly usage for running registry integration tests locally without cloud credentials.

## 2025-11-23 - 1.7.0 - feat(core)

Standardize S3 storage config using @tsclass/tsclass IS3Descriptor and wire it into RegistryStorage and plugins exports; update README and package dependencies.

- Add @tsclass/tsclass dependency to package.json to provide a standardized IS3Descriptor for S3 configuration.
- Export tsclass from ts/plugins.ts so plugin types are available to core modules.
- Update IStorageConfig to extend plugins.tsclass.storage.IS3Descriptor, consolidating storage configuration typing.
- Change RegistryStorage.init() to pass the storage config directly as an IS3Descriptor to SmartBucket (bucketName remains part of IStorageConfig).
- Update README storage section with example config and mention IS3Descriptor integration.

# 2025-11-21 - 1.6.0 - feat(core)

Add PyPI and RubyGems registries, integrate into SmartRegistry, extend storage and auth

- Introduce PyPI registry implementation with PEP 503 (Simple API) and PEP 691 (JSON API), legacy upload support, content negotiation and HTML/JSON generators (ts/pypi/\*).
- Introduce RubyGems registry implementation with Compact Index support, API v1 endpoints (upload, yank/unyank), versions/names files and helpers (ts/rubygems/\*).
- Wire PyPI and RubyGems into the main orchestrator: SmartRegistry now initializes, exposes and routes requests to pypi and rubygems handlers.
- Extend RegistryStorage with PyPI and RubyGems storage helpers (metadata, simple index, package files, compact index files, gem files).
- Extend AuthManager to support PyPI and RubyGems UUID token creation, validation and revocation and include them in unified token validation.
- Add verification of client-provided hashes during PyPI uploads (SHA256 always calculated and verified; MD5 and Blake2b verified when provided) to prevent corrupted uploads.
- Export new modules from library entry point (ts/index.ts) and add lightweight rubygems index file export.
- Add helper utilities for PyPI and RubyGems (name normalization, HTML generation, hash calculations, compact index generation/parsing).
- Update documentation hints/readme to reflect implementation status and configuration examples for pypi and rubygems.

# 2025-11-21 - 1.5.0 - feat(core)

Add PyPI and RubyGems protocol support, Cargo token management, and storage helpers

- Extend core protocol types to include 'pypi' and 'rubygems' and add protocol config entries for pypi and rubygems.
- Add PyPI storage methods for metadata, Simple API HTML/JSON indexes, package files, version listing and deletion in RegistryStorage.
- Add Cargo-specific storage helpers (index paths, crate storage) and ensure Cargo registry initialization and endpoints are wired into SmartRegistry.
- Extend AuthManager with Cargo, PyPI and RubyGems token creation, validation and revocation methods; update unified validateToken to check these token types.
- Update test helpers to create Cargo tokens and return cargoToken from registry setup.

# 2025-11-21 - 1.4.1 - fix(devcontainer)

Simplify devcontainer configuration and rename container image

- Rename Dev Container name to 'gitzone.universal' and set image to `mcr.microsoft.com/devcontainers/universal:4.0.1-noble`
- Remove large inline comments and example 'build'/'features' blocks to simplify the `devcontainer.json`

# 2025-11-21 - 1.4.0 - feat(registrystorage)

Add `deleteMavenMetadata` to `RegistryStorage` and update Maven DELETE test to expect 204 No Content

- Add `deleteMavenMetadata(groupId, artifactId)` to `RegistryStorage` to remove `maven-metadata.xml`.
- Update Maven test to assert 204 No Content for DELETE responses (previously expected 200).

# 2025-11-21 - 1.3.1 - fix(maven)

Pass request path to Maven checksum handler so checksum files are resolved correctly

- Call `handleChecksumRequest` with the full request path from `MavenRegistry.handleRequest`
- Allows `getChecksum` to extract the checksum filename from the URL and fetch the correct checksum file from storage
- Fixes 404s when requesting artifact checksum files (md5, sha1, sha256, sha512)

# 2025-11-21 - 1.3.0 - feat(core)

Add Cargo and Composer registries with storage, auth and helpers

- Add Cargo registry implementation (ts/cargo) including index, publish, download, yank/unyank and search handlers
- Add Composer registry implementation (ts/composer) including package upload/download, metadata, packages.json and helpers
- Extend RegistryStorage with Cargo and Composer-specific storage helpers and path conventions
- Extend AuthManager with Composer token creation/validation and unified token validation support
- Wire SmartRegistry to initialize and route requests to cargo and composer handlers
- Add adm-zip dependency and Composer ZIP parsing helpers (extractComposerJsonFromZip, sha1 calculation, version sorting)
- Add tests for Cargo index path calculation and config handling
- Export new modules from ts/index.ts and add module entry files for composer and cargo

## 2025-11-21 - 1.2.0 - feat(maven)

Add Maven registry protocol support (storage, auth, routing, interfaces, and exports)

- Add Maven protocol to core types (TRegistryProtocol) and IRegistryConfig
- SmartRegistry: initialize Maven registry when enabled, route requests to /maven, and expose it via getRegistry
- RegistryStorage: implement Maven storage helpers (get/put/delete artifact, metadata, list versions) and path helpers
- AuthManager: add UUID token creation/validation/revocation for Maven and integrate into unified validateToken/authorize flow
- New ts/maven module: exports, interfaces and helpers for Maven coordinates, metadata, and search results
- Add basic Cargo (crates.io) scaffolding: ts/cargo exports and Cargo interfaces
- Update top-level ts/index.ts and package exports to include Maven (and cargo) modules
- Tests/helpers updated to enable Maven in test registry and add Maven artifact/checksum helpers

## 2025-11-20 - 1.1.1 - fix(oci)

Improve OCI manifest permission response and tag handling: include WWW-Authenticate header on unauthorized manifest GETs, accept optional headers in manifest lookup, and persist tags as a unified tags.json mapping when pushing manifests.

- getManifest now accepts an optional headers parameter for better request context handling.

- Unauthorized GET manifest responses now include a WWW-Authenticate header with realm/service/scope to comply with OCI auth expectations.
- PUT manifest logic no longer writes individual tag objects; it updates a consolidated oci/tags/{repository}/tags.json mapping using getTagsData and putObject.
- Simplified tag update flow when pushing a manifest: tags[reference] = digest and persist tags.json.

## 2025-11-20 - 1.1.0 - feat(oci)

Support monolithic OCI blob uploads; add registry cleanup/destroy hooks; update tests and docs

- OCI: Add monolithic upload handling in handleUploadInit — accept digest + body, verify digest, store blob and return 201 with Docker-Content-Digest and Location
- OCI: Include Docker-Distribution-API-Version header in /v2/ version check response
- Lifecycle: Persist upload session cleanup timer and provide destroy() to clear timers in OciRegistry
- Orchestrator: Add destroy() to SmartRegistry to propagate cleanup to protocol handlers
- Tests: Ensure test suites call registry.destroy() in postTask cleanup to prevent leaked timers/resources
- Package metadata: bump @git.zone/tstest dev dependency and add packageManager field
- Docs: Readme formatting and legal/trademark/company information updated

## 2025-11-20 - 1.0.2 - fix(scripts)

Increase tstest timeout from 30s to 240s in package.json test script

- Extend the tstest CLI timeout in package.json from 30 seconds to 240 seconds to accommodate longer-running tests and reduce CI timeouts.

## 2025-11-20 - 1.0.1 - registry

Release 1.0.1 brings core registry features, multi-registry support, logging integration, and performance improvements to object listing. Also includes the initial project scaffold and CI/CD setup.

- Add smartlog dependency and integrate structured logging into the NpmRegistry class for better runtime diagnostics and traceability.
- Update smartbucket dependency to 4.3.0 and refactor listObjects for improved performance and efficiency when enumerating stored objects.

- Implement multi-registry support (multiple iterations/refinements included: v2, v3) to allow managing and resolving packages across multiple registries.
  - Initial project setup: TypeScript project scaffold, development tooling, and CI/CD workflows for automated testing and publishing.
    - Co-authored-by: Ona [no-reply@ona.com](mailto:no-reply@ona.com)
  - Miscellaneous housekeeping and small updates (2025-11-19 — 2025-11-20): several commits with no substantive messages; grouped as non-functional/maintenance changes.
- 

Revision #2

Created 2026-03-28 13:10:58 UTC by foss.global Team

Updated 2026-03-29 16:54:03 UTC by foss.global Team