

Polarion SCIM Interface

Architecture & Data Flow

CorbinSoft

Version 1.0 — June 2026

DRAFT — Pre-Development

Table of Contents

TOC \h \o "1-3"

1. Purpose and Scope

This document defines the architecture, component design, and data flows for the Polarion SCIM Interface — an OSGi bundle deployed directly into the Polarion ALM container. It covers three functional areas delivered across two phases:

- SCIM 2.0 Interface — standards-compliant user and group provisioning driven by an enterprise Identity Provider (IdP)
- Two-Phase Regulatory Provisioning — a training-gated onboarding workflow for regulated environments
- License Manager / Watchdog — license type management with an optional elevation and approval workflow

This document is intended for solution architects, developers, and technical reviewers. It is not normative implementation guidance; detailed design specifications will follow per subsystem.

2. Background and Context

2.1 SCIM 2.0 Standard

SCIM (System for Cross-domain Identity Management) is an open standard published as IETF RFC 7642, RFC 7643, and RFC 7644. It defines a REST API and JSON schema for automating user provisioning and de-provisioning across cloud and enterprise systems. Its design goals are simplicity, interoperability, and low integration cost.

The core object model derives all resources from a base Resource type carrying id, externalId, and meta. RFC 7643 defines User, Group, and EnterpriseUser as the standard resource types. The protocol (RFC 7644) maps these resources to standard HTTP verbs over a JSON+REST interface.

2.2 Polarion ALM

Polarion is a Siemens enterprise Application Lifecycle Management (ALM) platform. It runs on an OSGi container (Apache Felix) with an embedded Jetty web server. Polarion extensions are OSGi bundles that register against the container at runtime, gaining direct in-JVM access to Polarion's internal service APIs without requiring HTTP round-trips.

Relevant internal APIs include IUserService (user CRUD), IProjectService (project membership and roles), ISecurityService (permissions and license assignment), and IWorkItemService (work item and workflow management). All data persists in Polarion's own database.

2.3 Motivation

Enterprise customers deploying Polarion in regulated industries (medical devices, aerospace, automotive) require automated, auditable user lifecycle management. Manual provisioning is error-prone, slow, and inconsistent with compliance requirements. Integrating SCIM directly into Polarion as an OSGi bundle eliminates a separate middleware tier, reduces attack surface, and enables direct use of Polarion's internal APIs.

3. Design Principles

Principle	Rationale
Single deployment unit	The entire interface ships as one OSGi bundle. No external middleware, no separate process, no network hop to Polarion.
One authentication layer	Because the bundle runs inside Polarion, there is no outbound credential to manage. A single inbound bearer token gates all external requests.
Polarion-native persistence	All state (provisioning, license, activity) is stored in Polarion's own database using custom fields and work items. No external database is introduced.
Phased delivery	Each subsystem defines Phase 1 (minimal viable) and Phase 2 (full feature) scopes, allowing incremental delivery without architectural rework.
Abstraction at extension points	Interfaces are defined at approval and trigger boundaries so Phase 2 can substitute implementations (e.g., external ITSM approval) without restructuring core logic.
Principle of least privilege	The service account context used internally is scoped to the minimum permissions required for provisioning and license operations.

4. System Overview

4.1 Deployment Context

The Polarion SCIM Interface is deployed as a single OSGi bundle within the Polarion container. It registers HTTP endpoints against Polarion's embedded Jetty server and accesses Polarion's internal services via OSGi Declarative Services (DS) injection.

Two categories of external system interact with the bundle:

- SCIM Clients (e.g., Okta, Azure AD, OneLogin) — enterprise Identity Providers that drive user and group lifecycle events via the SCIM 2.0 protocol.
- Training Systems — LMS or course platforms that notify the bundle upon successful completion of mandatory training by a provisioned user.

All inbound requests, regardless of origin, pass through a single Bearer Token Auth Filter before reaching any business logic.

4.2 Subsystems

The bundle contains three logical subsystems:

Subsystem	Phase 1 Scope	Phase 2 Scope
SCIM Interface	Full SCIM 2.0: Users, Groups, Discovery, Bulk	No change — SCIM 2.0 is complete in Phase 1
Regulatory Provisioning	Two-phase onboarding: training-gated access grant	No change — fully delivered in Phase 1
License Manager	Manual license type changes with approval workflow and timed revert	Activity monitoring, utilization scoring, and automatic elevation/promotion

5. Authentication Architecture

5.1 Single Inbound Token

Because the OSGi bundle runs inside the Polarion JVM, there is no outbound authentication credential. The bundle calls Polarion's internal Java APIs directly, operating under a configured internal service account context. This eliminates the “two-leg” auth pattern that would be required by a standalone middleware service.

All external requests — whether from a SCIM client or a training system — must present a Bearer token in the Authorization header:

```
Authorization: Bearer <token>
```

5.2 Token Validation

Phase 1 uses a pre-shared static bearer token stored as a configuration property within the bundle. This approach minimizes infrastructure dependencies and is appropriate for initial deployment.

The token validation interface is designed to be replaceable. Phase 2 may substitute OAuth 2.0 token introspection (RFC 7662) or a JWT validation strategy without modifying the downstream business logic. The Auth Filter delegates to a pluggable TokenValidator interface.

5.3 Internal Service Account

All Polarion internal API calls execute under a dedicated service account configured in Polarion. This account:

- Is a non-human, non-personal account not tied to any individual
- Holds the minimum permissions required for user provisioning, project membership, and license management
- Is referenced by name in the bundle's configuration properties — not hardcoded
- Requires no PAT or bearer token because calls are made in-JVM, not over HTTP

6. SCIM Interface Subsystem

6.1 Endpoints

The SCIM interface exposes the following endpoints, all mounted under `/polarion/scim/v2/`:

Endpoint	Method(s)	Description
<code>/Users</code>	GET, POST	List/filter users; create a new user
<code>/Users/{id}</code>	GET, PUT, PATCH, DELETE	Read, replace, update, or delete a specific user
<code>/Groups</code>	GET, POST	List/filter groups; create a new group
<code>/Groups/{id}</code>	GET, PUT, PATCH, DELETE	Read, replace, update, or delete a specific group
<code>/Bulk</code>	POST	Execute multiple operations in a single request (RFC 7644 §3.7)
<code>/ServiceProviderConfig</code>	GET	Advertise supported features, auth schemes, and compliance
<code>/ResourceTypes</code>	GET	Enumerate available resource types
<code>/Schemas</code>	GET	Return full schema definitions for all resource types

6.2 SCIM-to-Polarion Mapping

SCIM User attributes map to Polarion `IUserService` fields as follows:

SCIM Attribute	Polarion Field	Notes
<code>userName</code>	<code>loginName</code>	Primary identifier; must be unique
<code>externalId</code>	<code>externalId</code> (custom field)	IdP-side identifier for reconciliation
<code>name.givenName</code>	<code>firstName</code>	
<code>name.familyName</code>	<code>lastName</code>	
<code>emails[primary]</code>	<code>email</code>	Primary email only
<code>active</code>	<code>disabled</code> (inverted)	<code>active=false</code> → Polarion <code>disabled=true</code>
<code>groups</code>	Project membership	Resolved via <code>IProjectService</code>

6.3 Data Flow — User Provisioning (SCIM Create)

The following sequence describes a SCIM POST `/Users` request creating a new user:

- SCIM client sends POST /polarion/scim/v2/Users with Authorization: Bearer <token> and a SCIM User JSON body.
- Bearer Token Auth Filter validates the token. Returns HTTP 401 if invalid.
- SCIM REST Layer deserializes the request body and validates required attributes (userName, name).
- Request is forwarded to the Provisioning Engine (see Section 7) which determines whether Phase 1 or full provisioning applies.
- Provisioning Engine calls IUserService.createUser() with mapped attributes.
- SCIM REST Layer serializes the created Polarion user back to a SCIM User response.
- HTTP 201 Created is returned with a Location header pointing to the new resource URI.

6.4 Data Flow — User Deprovisioning (SCIM Delete)

- SCIM client sends DELETE /polarion/scim/v2/Users/{id}.
- Auth Filter validates token.
- SCIM REST Layer resolves the Polarion user by SCIM id.
- IUserService.disableUser() is called (soft delete — account is disabled, not removed, to preserve audit trail).
- HTTP 204 No Content is returned.

Hard deletion is not supported in Phase 1 to preserve Polarion's audit and traceability records. This aligns with the soft-delete SCIM draft extension (draft-ansari-scim-soft-delete).

7. Regulatory Provisioning Subsystem

7.1 Overview

In regulated industries, granting a new user immediate access to production Polarion projects before completing mandatory training is a compliance violation. The Provisioning Engine implements a two-phase onboarding state machine to address this requirement.

User provisioning state is stored as a custom field on the Polarion user object and persists in Polarion's database. The two states are:

State	Meaning
PENDING_TRAINING	User account created. Login enabled. Access restricted to designated training projects only. Production projects not accessible.
ACTIVE	Training confirmed complete. Full role and project access granted per the user's provisioning profile.

7.2 Phase 1 — SCIM Create Trigger

When the SCIM Interface receives a Create User request:

- Provisioning Engine calls `IUserService.createUser()` with basic attributes.
- `IProjectService` grants membership to the configured training project(s) only.
- `ISecurityService` assigns a restricted role (e.g., Reviewer) appropriate for training access.
- User provisioning state is set to `PENDING_TRAINING` and persisted as a custom field.
- Provisioning timestamp is recorded for audit purposes.
- SCIM 201 Created response is returned to the calling IdP.

7.3 Phase 2 — Training Completion Trigger

The training system notifies the bundle upon successful course completion. The notification is delivered as an HTTP POST to a dedicated callback endpoint:

```
POST /polarion/scim/training/complete
```

The request body carries at minimum the user's `externalId` or `userName` and confirmation of training completion status. The same Bearer Token Auth Filter protects this endpoint.

Upon receiving a valid training completion notification:

- Provisioning Engine looks up the Polarion user by `externalId` or `userName`.

- Validates that user state is PENDING_TRAINING (idempotent — ACTIVE users are a no-op).
- IProjectService grants membership to production project(s) per the user's provisioning profile.
- ISecurityService updates the user's role to the fully provisioned role assignment.
- User provisioning state is updated to ACTIVE with a completion timestamp.
- HTTP 200 OK is returned to the training system.

7.4 Provisioning Profile

The set of training projects, production projects, and role assignments for each class of user is defined in a provisioning profile configuration. Profiles are stored as configuration properties in the bundle and reference Polarion project IDs and role IDs by name. This decouples the business rules from the code.

8. License Manager / Watchdog Subsystem

8.1 Overview

Polarion licenses are typed — different license types (e.g., Reviewer, Requirements Engineer, Full) grant different feature access. Managing these manually is operationally expensive and prone to over-licensing. The License Manager subsystem provides tooling to manage license types with controlled change management.

The subsystem is delivered in two phases with a clear scope boundary:

	Phase 1	Phase 2
Scope	Manual license type changes with request/approval workflow and timed auto-revert	Adds activity monitoring, per-user utilization scoring, and automatic elevation/demotion based on scoring
Trigger	User or manager initiates a request	System-generated signals from activity analysis also trigger changes
Persistence	Elevation records as Polarion work items	Adds activity logs and scoring data in Polarion database

8.2 Phase 1 — Elevation Handler

8.2.1 Request Flow

A user or manager initiates a license elevation request. In Phase 1, this is submitted via the bundle's REST API or directly as a Polarion work item (the mechanism is configurable):

- Requestor submits an elevation request specifying: target user, requested license type, justification, and requested duration.
- Elevation Handler creates a Polarion work item representing the request, routing it to the target user's manager via Polarion's work item workflow.
- Manager reviews and approves or rejects the work item within Polarion.
- On approval, a workflow transition fires an event that the Elevation Handler listens to.
- Elevation Handler calls `ISecurityService` to update the user's license type to the requested level.
- An expiry record is written (target user, original license type, expiry timestamp) to a custom field or work item for the revert scheduler.

8.2.2 Approval Abstraction

The approval mechanism is encapsulated behind an `ApprovalProvider` interface with a single method: `submitForApproval(ElevationRequest) → ApprovalHandle`. Phase 1 provides a `PolarionWorkItemApprovalProvider`. Phase 2 (or a subsequent delivery) may provide an `ExternalTSMApprovalProvider` (e.g., ServiceNow, Jira Service Management) by implementing the same interface, without modifying the Elevation Handler.

8.2.3 Timed Revert

The OSGi Scheduler runs a periodic job (configurable interval, default daily) that:

- Queries all active elevation records for those past their expiry timestamp.
- For each expired elevation, calls ISecurityService to revert the user's license type to the original pre-elevation value.
- Updates the elevation record status to EXPIRED.
- Optionally notifies the user and manager via Polarion notification mechanisms.

8.3 Phase 2 — Activity Monitoring and Scoring (Future)

Phase 2 extends the License Manager with three additional components. These are described here for architectural completeness but are not in scope for initial delivery.

- **Activity Collector:** Registers Polarion event listeners to capture login events, session duration, and feature-specific interactions. Writes activity records to Polarion's database.
- **Utilization Analyzer:** A scheduled OSGi job that aggregates activity data per user over a rolling window and computes a utilization score against the user's current license type. Flags users whose usage does not justify their license level.
- **Automated Elevation/Demotion:** The Elevation Handler gains a second trigger path — in addition to manual requests, the Analyzer can programmatically submit elevation or demotion recommendations. The same ApprovalProvider interface governs whether manager approval is required for automated actions.

9. OSGi Bundle Structure

9.1 Key Packages

Package	Contents
com.corbinsoft.polarion.scim.filter	Bearer Token Auth Filter — intercepts all inbound requests
com.corbinsoft.polarion.scim.rest	SCIM REST Layer — servlet registration, JSON marshaling, routing
com.corbinsoft.polarion.scim.provisioning	Provisioning Engine — Phase 1 and Phase 2 logic, state machine
com.corbinsoft.polarion.scim.training	Training Callback Handler — endpoint and event processing
com.corbinsoft.polarion.scim.license	License Manager — Elevation Handler, ApprovalProvider interface, Scheduler job
com.corbinsoft.polarion.scim.config	Configuration reader — bundle properties, provisioning profiles
com.corbinsoft.polarion.scim.model	SCIM domain model — User, Group, ListResponse, ErrorResponse

9.2 Polarion Internal APIs Consumed

Polarion API	Used By
IUserService	SCIM REST Layer, Provisioning Engine — create, update, disable users
IProjectService	Provisioning Engine — training and production project membership
ISecurityService	Provisioning Engine, Elevation Handler — role assignment, license type management
IWorkItemService	Elevation Handler — create and monitor approval work items
IDataService / Custom Fields	Provisioning Engine, Elevation Handler — provisioning state, elevation records
OSGi HttpService / Jetty	SCIM REST Layer, Training Callback Handler — servlet registration
OSGi Scheduler	License Manager — periodic elevation expiry checks and (Phase 2) activity analysis

9.3 Build and Deployment

The bundle is built with Maven using the Felix Maven Bundle Plugin (maven-bundle-plugin) to generate OSGi MANIFEST.MF headers. Polarion JARs are declared with provided scope. The built JAR is dropped into the polarion/extensions/ directory of the Polarion installation. No application server restart is required beyond the standard Polarion bundle activation cycle.

10. Delivery Phases Summary

Component	Phase 1	Phase 2
SCIM 2.0 REST API	☑ Full (Users, Groups, Bulk, Discovery)	— No change
Bearer Token Auth Filter	☑ Static pre-shared token	⬆ Optional: OAuth 2.0 introspection or JWT
Two-Phase Provisioning	☑ Full (Phase 1 + Phase 2 onboarding)	— No change
Training Callback Endpoint	☑ Full	— No change
License Elevation Handler	☑ Manual request + Polarion WI approval + timed revert	⬆ Adds automated trigger from activity scoring
Approval Provider (external)	✗ Not in scope	☑ ExternalITSMApprovalProvider (ServiceNow / Jira)
Activity Collector	✗ Not in scope	☑ Polarion event listeners for login/session/features
Utilization Analyzer	✗ Not in scope	☑ Scheduled scoring job, underuse flagging

11. Open Items

The following items require resolution before or during detailed design:

#	Item	Owner	Phase
1	Confirm Polarion version range and available internal API surface (IUserService signature differences across versions)	TBD	1
2	Define provisioning profile schema — training project IDs, production project IDs, and role mappings per user class	TBD	1
3	Specify training completion callback payload contract with the training system team	TBD	1
4	Define bearer token distribution and rotation procedure for SCIM clients	TBD	1
5	Confirm Polarion license type identifiers and ISecurityService API for license assignment	TBD	1
6	Identify external ITSM system for Phase 2 approval integration (ServiceNow, Jira SM, other)	TBD	2
7	Define activity scoring model — metrics, weights, thresholds for underuse classification	TBD	2

Appendix A — SCIM 2.0 Reference

The following RFCs govern the SCIM 2.0 standard implemented by this interface:

- RFC 7642 — SCIM: Definitions, Overview, Concepts, and Requirements
- RFC 7643 — SCIM: Core Schema (User, Group, EnterpriseUser, common attributes)
- RFC 7644 — SCIM: Protocol (REST API, HTTP bindings, error handling, bulk operations, filtering)
- RFC 6750 — OAuth 2.0 Bearer Token Usage (governs Authorization header format)

All SCIM endpoints return Content-Type: application/scim+json. Filter syntax, sorting, and attribute projection follow RFC 7644 Section 3.4.

Appendix B — Glossary

Term	Definition
SCIM	System for Cross-domain Identity Management. IETF open standard for user provisioning APIs.
IdP	Identity Provider. System that manages user identities and drives provisioning events (e.g., Okta, Azure AD).
OSGi	Open Services Gateway initiative. Component framework used by Polarion (Apache Felix runtime).
PAT	Personal Access Token. Not used in this architecture — replaced by in-JVM service account context.
Elevation	Temporary assignment of a higher-tier Polarion license type to a user for a bounded period.
Provisioning Profile	Configuration artifact defining project memberships and role assignments for a class of user.
ApprovalProvider	Interface abstracting the approval mechanism for license elevation requests. Pluggable per phase.