WebhookService

September 4, 2026

WebhookService

Kind: Service

Source: `atloria-monorepo/apps/api/src/webhooks/webhook.service.ts`

`WebhookService` processes incoming repository webhooks from GitHub, GitLab, and Azure DevOps. It validates provider-specific authentication, resolves the affected project and trigger configuration, and dispatches tag, push, or release events into background jobs when configured.

Methods

Method	Signature	Returns	Description
`verifyGitHubSignature`	`verifyGitHubSignature(signature: string, body: string | Buffer, secret: string)`	`void`	
`verifyGitLabToken`	`verifyGitLabToken(token: string, secret: string)`	`void`	Verify GitLab webhook token.
`verifyAzureDevOpsAuth`	`verifyAzureDevOpsAuth(authHeader: string, secret: string)`	`void`	Verify Azure DevOps Basic Auth header.
`findProjectByRepoUrl`	`findProjectByRepoUrl(repoUrl: string, event: 'push' | 'tag_create' | 'release')`	`unknown`	Get trigger config for a project.
`handleTagCreate`	`handleTagCreate(repoUrl: string, tagName: string, commitHash: string, provider: string,`	`Promise<{ status: string;`	Handle tag creation event.

Method	Signature	Returns	Description
	`rawPayload: any, verify: WebhookVerification)`	`jobId?: string }>`	
`handlePush`	`handlePush(repoUrl: string, branch: string, commitHash: string, provider: string, rawPayload: any, verify: WebhookVerification)`	`Promise<{ status: string; jobId?: string }>`	Handle push event.
`handleRelease`	`handleRelease(repoUrl: string, tagName: string, releaseName: string, commitHash: string, provider: string, rawPayload: any, verify: WebhookVerification)`	`Promise<{ status: string; jobId?: string }>`	Handle release event.
`generateWebhookSecret`	`generateWebhookSecret()`	`string`	Generate a new webhook secret.

Dependencies

- `PrismaService`
- `DocAutomationService`
- `TechnicalDocsService`
- `DocsPrTrigger` *(optional)*
- `GithubAppService` *(optional)*

Where it refuses work

- `WebhookService` stops the work with `ForbiddenException` when `!signature || !secret` — “Missing webhook signature or secret”.
- `WebhookService` stops the work with `ForbiddenException` when `sigBuf.length !== expBuf.length || !crypto.timingSafeEqual(sigBuf, expBuf)` — “Invalid webhook signature”.
- `WebhookService` stops the work with `ForbiddenException` when `!token || !this.timingSafeStrEq(token, secret)` — “Invalid GitLab webhook token”.
- `WebhookService` stops the work with `ForbiddenException` when `!authHeader || !secret` — “Missing Azure DevOps auth or secret”.

- `WebhookService` stops the work with `ForbiddenException` when `!this.timingSafeStrEq(authHeader, expected)` — “Invalid Azure DevOps credentials”.
- `WebhookService` stops the work with `ForbiddenException` when `!secret` — “Webhook secret is not configured for this project.”.

When something fails

- `WebhookService` handles failure in 3 places: it turns it into a return value in 2, and logs it and continues in 1.

Diagram

```
sequenceDiagram
  participant Provider as Git Provider
  participant Controller as Webhook Controller
  participant Service as WebhookService
  participant Projects as Project Store
  participant Jobs as Job Queue

  Provider->>Controller: POST webhook event
  Controller->>Service: Verify provider signature/token

  alt GitHub
    Service->>Service: verifyGitHubSignature()
  else GitLab
    Service->>Service: verifyGitLabToken()
  else Azure DevOps
    Service->>Service: verifyAzureDevOpsAuth()
  end

  Controller->>Service: Handle event payload
  Service->>Projects: findProjectByRepoUrl(repoUrl)
  Service->>Projects: getTriggerConfig(project, event)

  alt Tag creation
    Service->>Service: handleTagCreate()
  else Push event
    Service->>Service: handlePush()
  else Release event
    Service->>Service: handleRelease()
  end

  Service->>Jobs: Enqueue configured job
  Jobs-->>Service: jobId
  Service-->>Controller: { status, jobId? }
  Controller-->>Provider: Webhook response
```

Usage

```
import { Injectable } from '@nestjs/common';
import { WebhookService } from './webhook.service';

@Injectable()
export class GitHubWebhookHandler {
  constructor(private readonly webhookService: WebhookService) {}

  async handlePushWebhook(
    payload: Record<string, unknown>,
    signature: string,
    rawBody: Buffer,
  ) {
    this.webhookService.verifyGitHubSignature(rawBody, signature);

    return this.webhookService.handlePush(payload);
    // Example result:
    // { status: 'queued', jobId: 'job_123' }
  }
}

// Generate and persist a provider webhook secret during project setup.
const secret = webhookService.generateWebhookSecret();
```

AI Coding Instructions

- Always validate the provider signature, token, or authorization header before reading event data or scheduling jobs.
- Resolve projects using the canonical repository URL; normalize provider-specific URL formats consistently before calling `findProjectByRepoUrl()`.
- Route webhook payloads to `handleTagCreate()`, `handlePush()`, or `handleRelease()` based on the provider event type rather than inferred payload fields alone.
- Preserve the `{ status, jobId? }` response contract so controllers can return predictable acknowledgements to webhook providers.
- Use `generateWebhookSecret()` for new integrations and store the result securely; never log webhook secrets, signatures, or authorization tokens.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `DocAutomationService`
- DEPENDS_ON → `TechnicalDocsService`

- DEPENDS_ON → `DocsPrTrigger`
- DEPENDS_ON → `GithubAppService`

Referenced By

- `GithubAppController` (DEPENDS_ON)
- `WebhookController` (DEPENDS_ON)
- `WebhookModule` (MODULE_PROVIDES)
- `WebhookModule` (MODULE_EXPORTS)