

Docs » Symfony Bundles » Platform Bundle » Apps (plugin system)

Apps (plugin system)

Plugin/integration ecosystem with per-user, per-organization, and per-resource installations.

Anonymous · 22/09/2026

Apps (plugin system)

The Apps module provides a plugin/integration ecosystem for your SaaS. Apps are integrations or capabilities that users and organizations can enable — payment gateways, messaging platforms, storage services, email providers, and any custom capability your domain requires.

Concepts

Concept	Description
App	A registered integration (e.g. Stripe, Telegram, Dropbox).
Scope	Where an app can be installed: <code>user</code> , <code>organization</code> , or <code>resource</code> .
Installation	An app enabled for a specific user, org, or resource with config.
Capability	An interface an app can implement (e.g. payment, storage).
Catalog override	YAML configuration to adjust app metadata.

Defining an app

Create a service implementing `AppDefinitionInterface` (extend `AbstractApp` for convenience):

```
use Derafu\PlatformBundle\Apps\Abstract\AbstractApp;
use Derafu\PlatformBundle\Apps\Enum\AppScope;

final class StripeApp extends AbstractApp
{
    protected string $code = 'stripe';
    protected string $name = 'Stripe';
    protected string $description = 'Accept payments via Stripe.';
    protected string $icon = 'stripe';
    protected string $category = 'payments';

    protected array $scopes = [AppScope::Organization];

    // Keys to encrypt in the installation config
    protected array $sensitiveConfigKeys = ['secret_key', 'webhook_secret'];
}
```

```
// Optional: a form key for the configuration form
protected ?string $configFormKey = 'stripe_config';
}
```

The service is auto-discovered via the `AppDefinitionInterface` `AutoconfigureTag`. No manual service registration needed.

Scopes

An app can support one or more scopes:

Scope	Entity	Use case
<code>AppScope::User</code>	<code>BaseUserAppInstallation</code>	Personal integration (e.g. Telegram notifications)
<code>AppScope::Organization</code>	<code>BaseOrganizationAppInstallation</code>	Shared integration for the whole org
<code>AppScope::Resource</code>	<code>BaseResourceAppInstallation</code>	Integration for a specific resource

A multi-scope app (e.g. a storage app) can be installed at the user, organization, and/or resource level simultaneously.

Installing and managing apps

```
use Derafu\PlatformBundle\Apps\Contract\Service\AppInstallationManagerInterface;

// Install for a user
$installation = $appInstallationManager->installForUser(
    appCode: 'stripe',
    user: $currentUser,
    config: ['public_key' => 'pk_live...', 'secret_key' => 'sk_live...']
);

// Install for an organization
$installation = $appInstallationManager->installForOrganization(
    appCode: 'stripe',
    organization: $org,
    config: ['public_key' => 'pk_live...', 'secret_key' => 'sk_live...']
);

// Check if installed
$installed = $appInstallationManager->isInstalledForOrganization('stripe', $org);

// Get installation
$installation = $appInstallationManager->getForOrganization('stripe', $org);
```

```
// Uninstall
$appInstallationManager->uninstallForOrganization('stripe', $org);
```

Sensitive config keys declared in `$sensitiveConfigKeys` are transparently encrypted at rest by `AppConfigEncryptor`. Retrieval also decrypts transparently.

The app registry

The `AppRegistryInterface` collects all registered apps. It applies catalog overrides from configuration (scopes, flags, tags, supported resources):

```
use Derafu\PlatformBundle\Apps\Contract\Service\AppRegistryInterface;

$app = $appRegistry->get('stripe');           // AppDefinitionInterface
$app = $appRegistry->all();                     // all registered apps
$app = $appRegistry->byScope(AppScope::User); // filtered by scope
$app = $appRegistry->byCategory('payments'); // filtered by category
```

Catalog configuration

The YAML catalog section adjusts metadata of registered apps without modifying the app class. Useful for overriding defaults per-deployment:

```
derafu_platform:
  apps:
    catalog:
      stripe:
        scopes: [user, organization]
        flags: [featured]
      telegram:
        scopes: [user]
        flags: [featured]
      dropbox:
        scopes: [user, organization, resource]
        resources:
          - App\Entity\Resource\Project
```

`flags` controls UI presentation (e.g. featured items appear prominently). `resources` lists which FQCN resource classes the app supports when installed at the resource scope.

Capabilities

Apps can implement capability interfaces to advertise features. The `CapabilityRegistryInterface`

answers “which installed apps for this organization support X?”:

```
use Derafu\PlatformBundle\Apps\Contract\Capability\PaymentGatewayInterface;

// Define a capability interface (app side)
interface PaymentGatewayInterface
{
    public function processPayment(int $amountCents, string $currency): string;
}

// An app implements it
final class StripeApp extends AbstractApp implements PaymentGatewayInterface
{
    public function processPayment(int $amountCents, string $currency): string
    {
        // Stripe API call...
    }
}

// Query at runtime
use Derafu\PlatformBundle\Apps\Contract\Service\CapabilityRegistryInterface;

$gateways = $capabilityRegistry->forOrganization(PaymentGatewayInterface::class,
$org);
// Returns all installed apps for $org that implement PaymentGatewayInterface
```

Resource apps

For apps that target a specific resource (e.g. “email provider for this Project”), create a resource installation entity:

```
use Derafu\PlatformBundle\Apps\Entity\BaseResourceAppInstallation;

#[ORM\Entity]
#[ORM\Table(name: 'project_app_installations')]
class ProjectAppInstallation extends BaseResourceAppInstallation
{
    #[ORM\ManyToOne(targetEntity: Project::class)]
    #[ORM\JoinColumn(nullable: false, onDelete: 'CASCADE')]
    private Project $project;

    public function getProject(): Project { return $this->project; }
    public function setProject(Project $project): static { $this->project = $project; }
    return $this; }
}
```

Settings UI

The Apps module registers a settings section automatically. Users and org admins can browse, install, configure, and uninstall apps from a dedicated settings page — no custom controllers needed.

Last updated on 22/09/2026