Identity UI

Built-in web UI for authentication and account management.

Anonymous · 09/11/2026

Identity UI

The IdentityUi module provides ready-made Symfony controllers, routes, and Twig templates for all authentication flows and account settings pages. You get a fully functional user-facing UI with minimal configuration.

What’s included

Authentication

Route name	Path	What it does
`derafu_platform_identity_login`	`/login`	Login form
`derafu_platform_identity_register`	`/register`	Registration form
`derafu_platform_identity_logout`	`/logout`	Logout
`derafu_platform_identity_password_reset_request`	`/password-reset`	Forgot password
`derafu_platform_identity_password_reset`	`/password-reset/{token}`	Reset password
`derafu_platform_identity_magic_link_request`	`/magic-link`	Request magic link
`derafu_platform_identity_verify_email_required`	`/verify-email/required`	“Please verify your email” wall
`derafu_platform_identity_sudo`	`/sudo`	Re-authentication for sudo mode
`2fa_login`	`/2fa/check`	2FA code entry

Account settings

Section	What it manages
Account	Display name, avatar, locale, timezone
Security	Password change, 2FA enrollment, active sessions, login history
Email	Email change with verification
API Keys	Create, list, revoke API keys

Section	What it manages
JWT Tokens	List and revoke JWT tokens
Organizations	Create orgs, manage members, invitations
Teams	Create and manage teams within organizations

Webhook settings (Notifications module)

When the Notifications module is active, a Webhooks section appears in settings for managing per-user and per-organization webhook endpoints.

App settings (Apps module)

When the Apps module is active, an Apps section appears for installing and configuring integrations.

Configuration

```
# config/packages/derafu_platform.yaml
derafu_platform:
    identity_ui:
        # Your app's base Twig layout that the bundle's templates extend.
        parent_layout: 'layouts/app.html.twig'

        # Sender address for transactional emails (verification, reset, etc.)
        from_email: '%env(MAILER_FROM_EMAIL)%'
        from_name:  '%env(MAILER_FROM_NAME)%'

        # When true, users who haven't verified their email are redirected
        # to the verification wall on every request.
        require_email_verification: true
```

Extending the parent layout

Your `parent_layout` template must define the blocks that the bundle's templates fill in. At minimum:

```
{# templates/layouts/app.html.twig #}
<!DOCTYPE html>
<html>
<head>
    <title>{% block title %}My App{% endblock %}</title>
    {% block stylesheets %}{% endblock %}
</head>
<body>
    {% block body %}{% endblock %}
```

```
    {% block javascripts %}{% endblock %}
</body>
</html>
```

The bundle's templates use `{% extends derafu_platform_parent_layout %}`, which resolves to your configured `parent_layout`.

Overriding templates

Override any template by creating a file at the same path under your app's `templates/` directory using Symfony's standard bundle template override:

```
templates/bundles/PlatformBundle/IdentityUi/page/auth/login.html.twig
```

Any template the bundle ships can be overridden this way — the bundle's templates are a starting point, not a constraint.

Locale negotiation

The `LocaleNegotiationListener` automatically negotiates the user's preferred locale on each request, matching against the `supported_locales` list. It checks (in order):

1. The user's saved locale preference (if authenticated).
2. The browser's `Accept-Language` header.
3. The Symfony default locale.

```
derafu_platform:
    identity:
        supported_locales:
            en: English
            es: Español
            fr: Français
```

Email verification wall

When `require_email_verification: true`, any authenticated user whose email is not yet verified is redirected to `derafu_platform_identity_verify_email_required` on every request. This enforcer runs as a `kernel.request` listener and can be scoped to specific firewalls.

A "Resend verification email" button is available on that page.

Settings section providers

The settings UI is assembled from registered section providers. Any bundle or module can contribute a settings section by implementing `SettingsSectionProviderInterface`:

```
use Derafu\PlatformBundle\Core\Contract\Service\SettingsSectionProviderInterface;
use Symfony\Component\DependencyInjection\Attribute\AutoconfigureTag;

#[AutoconfigureTag('derafu.settings_section_provider')]
final class BillingSettingsSection implements SettingsSectionProviderInterface
{
    public function getSection(): array
    {
        return [
            'code'     => 'billing',
            'label'    => 'Billing',
            'icon'     => 'credit-card',
            'route'    => 'app_settings_billing',
            'priority' => 50,
        ];
    }
}
```

The `SettingsExtension` Twig extension collects all registered sections and renders them in the settings sidebar.

Email delivery

The bundle uses Symfony Mailer for all transactional emails (verification, password reset, invitation, etc.). Configure your mailer transport in `config/packages/mailer.yaml`. The `from_email` and `from_name` values in `identity_ui` configuration are used as the sender on all emails.

Emails are triggered by subscribing to the bundle's events:

```
#[AsEventListener]
class IdentityMailerListener
{
    public function __invoke(UserRegistered $event): void
    {
        // Bundle's built-in IdentityMailerListener already handles this
        // if you leave it registered. Override it by replacing the alias.
    }
}
```

The bundle's built-in `IdentityMailerListener` handles all standard emails automatically. You can override individual email templates or swap the entire listener.

Last updated on 09/11/2026