User lifecycle

Registration, email verification, password reset, email change, and more.

Anonymous · 09/09/2026

User lifecycle

The bundle provides services for all common user lifecycle flows. Each service emits events so the app can send emails, log activity, or trigger side effects — the bundle stays transport-agnostic.

Token security model

All tokens (email verification, password reset, email change, magic link) share the same security model:

- **256 bits of entropy**: `bin2hex(random_bytes(32))` → 64-char hex string.
- **SHA-256 hash stored**: the database holds `hash('sha256', $plaintext)`, never the plaintext. A leaked database cannot be replayed.
- **Single-use**: consumption sets `usedAt`.
- **Time-boxed**: each token type has a configurable TTL.
- **`hash_equals()`**: timing-safe comparison on validation.

The plaintext is returned exactly once in a `GeneratedToken` wrapper and is meant to be emailed immediately.

Registration

```
use Derafu\PlatformBundle\Identity\Contract\Service\RegistrarInterface;

class RegisterController
{
    public function __invoke(RegistrarInterface $registrar, Request $request):
Response
    {
        $registration = $registrar->register(
            email: 'alice@example.com',
            plainPassword: 'secret',
            name: 'Alice'
        );

        // $registration->user                                  — the persisted User
```

```
        // $registration->verificationToken->plaintext     — for the email

        return new JsonResponse(['message' => 'Check your inbox.'], 201);
    }
}
```

The `UserRegistered` event is dispatched with the plaintext token. Subscribe to it to send the confirmation email:

```
#[AsEventListener]
class SendWelcomeEmail
{
    public function __invoke(UserRegistered $event): void
    {
        // Build URL: /verify-email?token={$event->verificationTokenPlaintext}
        // Send email to $event->user->getEmail()
    }
}
```

Email verification

```
use Derafu\PlatformBundle\Identity\Contract\Service\EmailVerifierInterface;

// GET /verify-email?token=abc123...
$user = $emailVerifier->verify($request->query->get('token'));
// $user->isEmailVerified() is now true
```

Emits `UserEmailVerified`.

Password reset

Request phase (user submits email):

```
use Derafu\PlatformBundle\Identity\Contract\Service\PasswordResetterInterface;

$passwordResetter->requestReset('alice@example.com');
// Silent no-op if email not registered (anti-enumeration).
```

Emits `PasswordResetRequested` (only when user exists). Subscribe to send the reset email with the plaintext token.

Reset phase (user clicks link, submits new password):

```
$user = $passwordResetter->reset($token, $newPassword);
```

Emits `UserPasswordChanged`.

Email change

```
use Derafu\PlatformBundle\Identity\Contract\Service\EmailChangerInterface;

// Step 1: request (sends confirmation to the NEW email)
$generated = $emailChanger->requestChange($currentUser, 'newemail@example.com');

// Step 2: confirm (user clicks link in the new email)
$user = $emailChanger->confirmChange($token);
// $user->getEmail() is now 'newemail@example.com'
```

The new email is bound to the token's payload — it cannot be swapped after issuance. Emits `EmailChangeRequested` (step 1) and `UserEmailChanged` (step 2, carries both old and new addresses).

Magic links (passwordless login)

Magic links let users log in by clicking a link sent to their email — no password needed.

Request a link:

```
use Derafu\PlatformBundle\Identity\Contract\Service\MagicLinkManagerInterface;

// POST /magic-link (user submits their email)
$magicLinkManager->requestLink('alice@example.com');
// Silent no-op if email not registered (anti-enumeration).
```

Emits `MagicLinkRequested` (only when user exists). Subscribe to send the login email:

```
#[AsEventListener]
class SendMagicLinkEmail
{
    public function __invoke(MagicLinkRequested $event): void
    {
        $url = 'https://app.example.com/magic-link?_magic_link_token='
             . $event->plaintextToken;
        // Send email to $event->user->getEmail() with $url
    }
}
```

Authenticate via the link:

The bundle ships a Symfony Security authenticator. Enable it in `security.yaml`:

```
security:
    firewalls:
        main:
            custom_authenticators:
                - Derafu\PlatformBundle\Identity\Security\MagicLinkAuthenticator
```

The authenticator fires on any request with a `_magic_link_token` query parameter, consumes the token, and authenticates the user.

Token cleanup

Expired tokens pile up over time. Run the purge command periodically:

```
bin/console identity:tokens:purge
```

Typically as a daily cron job.

Sudo mode (re-authentication for sensitive actions)

Sudo mode is a time-limited elevated state that proves the user recently confirmed their password. Actions that require sudo will throw `SudoRequiredException` when the window expires.

Confirm password:

```
use Derafu\PlatformBundle\Identity\Contract\Service\SudoManagerInterface;

$sudoManager->confirmPassword($this->getUser(), $request->get('password'));
// Sudo window refreshed.
```

Check sudo state:

```
$sudoManager->isGranted($user); // true within the TTL window
```

Auto-granted on login. The `SudoOnLoginListener` calls `grant()` after every successful login — users are not asked to re-confirm immediately after typing their password.

Session-based, no DB. The timestamp lives in the session. Works naturally in PHP-FPM.

Account lockout

After N consecutive failed login attempts (default 5), the account is automatically locked for a configurable duration (default 30 min).

Fully automatic. The `AccountLockListener` hooks into Symfony's authentication events — no app-side wiring needed.

Admin unlock:

```
use Derafu\PlatformBundle\Identity\Contract\Service\AccountLockManagerInterface;

$lockManager->unlock($user);
// Resets counter + clears lock. Emits AccountUnlocked.
```

Check programmatically:

```
$lockManager->isLocked($user); // true when lockedUntil > now
```

Login history

Every successful login is automatically recorded — IP, User-Agent, and which authenticator succeeded. Zero wiring needed.

Query recent logins:

```
use Derafu\PlatformBundle\Identity\Contract\Service\LoginHistoryManagerInterface;

$records = $loginHistoryManager->getRecentForUser($user, limit: 10);

foreach ($records as $record) {
    echo $record->getIpAddress();
    echo $record->getUserAgent();
    echo $record->getAuthenticator();
    echo $record->getCreatedAt()->format('Y-m-d H:i');
}
```

Purge old records:

```
bin/console identity:login-history:purge --days=90
```

TTL & lockout configuration

```
derafu_platform:
    identity:
        sudo_ttl: 900               # 15 minutes (default)
        lockout_max_attempts: 5     # lock after 5 failures (default)
        lockout_duration: 1800      # lock for 30 minutes (default)
        verification_ttl:
```

```
            email_verify: 86400     # 1 day (default)
            password_reset: 3600    # 1 hour (default)
            email_change: 86400     # 1 day (default)
            magic_link: 900         # 15 minutes (default)
```

Last updated on 09/09/2026