Events

All dispatched events and how to subscribe to them.

Anonymous · 09/09/2026

Events

All identity events live under `Derafu\PlatformBundle\Identity\Event\` and extend `Symfony\Contracts\EventDispatcher\Event`. The bundle dispatches them after successful operations — subscribe to them to send emails, log activity, seed data, or trigger side effects.

Subscribing

```
use Derafu\PlatformBundle\Identity\Event\UserRegistered;
use Symfony\Component\EventDispatcher\Attribute\AsEventListener;

#[AsEventListener]
class SendConfirmationEmail
{
    public function __invoke(UserRegistered $event): void
    {
        $user = $event->user;
        $token = $event->verificationTokenPlaintext;

        // Build URL and send email...
    }
}
```

User lifecycle events

Event	When	Key fields
`UserRegistered`	After `Registrar::register()`	`user`, `verificationTokenPlaintext`
`UserEmailVerified`	After `EmailVerifier::verify()`	`user`
`PasswordResetRequested`	After `PasswordResetter::requestReset()` (only if user exists)	`user`, `verificationTokenPlaintext`
`UserPasswordChanged`	After `PasswordResetter::reset()`	`user`
`EmailChangeRequested`	After `EmailChanger::requestChange()`	`user`, `newEmail`, `verificationTokenPlaintext`

Event	When	Key fields
`UserEmailChanged`	After `EmailChanger::confirmChange()`	`user`, `oldEmail`, `newEmail`
`MagicLinkRequested`	After `MagicLinkManager::requestLink()` (only if user exists)	`user`, `plaintextToken`
`AccountLocked`	After `AccountLockManager::handleFailedLogin()` when threshold reached	`user`, `lockedUntil`, `failedAttempts`
`AccountUnlocked`	After `AccountLockManager::unlock()`	`user`

Note on `PasswordResetRequested` and `MagicLinkRequested`: not dispatched when the email does not belong to any user — this is intentional to avoid leaking account existence.

Note on `UserEmailChanged`: carries the `oldEmail` so a listener can notify the old address ("if this wasn't you, contact support").

Organization events

Event	When	Key fields
`OrganizationCreated`	After `OrganizationManager::create()`	`organization`, `owner`
`OrganizationMemberAdded`	After `OrganizationManager::addMember()` or invitation accept	`membership`
`OrganizationMemberRemoved`	After `OrganizationManager::removeMember()`	`organization`, `user`
`OrganizationOwnershipTransferred`	After `OrganizationManager::transferOwnership()`	`organization`, `previousOwner`, `newOwner`

Note on `OrganizationMemberRemoved`: the membership entity is already deleted when listeners run — the event carries `organization` and `user` separately.

Invitation events

Event	When	Key fields
`InvitationCreated`	After `InvitationManager::invite()`	`invitation`, `plaintextToken`
`InvitationAccepted`	After `InvitationManager::accept()`	`invitation`, `membership`
`InvitationRevoked`	After `InvitationManager::revoke()`	`invitation`

Note on `InvitationCreated`: the `plaintextToken` is the only chance to build the accept URL. Once the request ends, only the hash in the database survives.

Team events

Event	When	Key fields
`TeamCreated`	After `TeamManager::create()`	`team`
`TeamMemberAdded`	After `TeamManager::addMember()`	`team`, `user`
`TeamMemberRemoved`	After `TeamManager::removeMember()`	`team`, `user`

Events that carry plaintext tokens

These events expose a plaintext token that **must not be logged or persisted**. They exist solely so the app's mailer can build a URL and send it:

- `UserRegistered.verificationTokenPlaintext`
- `PasswordResetRequested.verificationTokenPlaintext`
- `EmailChangeRequested.verificationTokenPlaintext`
- `MagicLinkRequested.plaintextToken`
- `InvitationCreated.plaintextToken`

Last updated on 09/09/2026