Organizations & invitations

Multi-tenant orgs, memberships, teams, invitations, and context.

Anonymous · 09/09/2026

Organizations & invitations

Creating an organization

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

class CreateOrgController
{
    public function __invoke(
        OrganizationManagerInterface $orgManager,
        RoleRepository $roleRepo
    ): Response {
        $ownerRole = $roleRepo->findOneBy(['code' => 'org.owner']);

        $org = $orgManager->create(
            owner: $this->getUser(),
            name: 'Acme Inc',
            ownerRole: $ownerRole
        );

        // $org is persisted with a membership (role = org.owner)
        return new JsonResponse(['id' => $org->getId()]);
    }
}
```

`create()` is transactional: the organization and the owner-membership are persisted atomically. Emits `OrganizationCreated`.

Adding members directly

```
$membership = $orgManager->addMember($org, $user, $memberRole);
```

Throws `OrganizationOperationException` if the user is already a member. Emits `OrganizationMemberAdded`.

Removing members

```
$orgManager->removeMember($membership);
```

Throws `OrganizationOperationException` if the membership is the owner (`role.code = 'org.owner'`). Transfer ownership first. Emits `OrganizationMemberRemoved`.

Transferring ownership

```
$adminRole = $roleRepo->findOneBy(['code' => 'org.admin']);

$orgManager->transferOwnership($org, $newOwner, demoteCurrentOwnerTo: $adminRole);
```

The new owner must already be a member. Both role changes happen in a single transaction. Emits `OrganizationOwnershipTransferred` with the previous and new owners.

Finding the owner

```
$owner = $orgManager->getOwner($org); // ?UserInterface
```

Scans memberships for `role.code = 'org.owner'`. Returns `null` if none found (should not happen if the invariant is maintained).

Teams

Teams are named groups of organization members. They simplify resource access grants: instead of adding 20 users individually to a resource, assign a team with a role.

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

// Create a team
$team = $teamManager->create($org, 'backend', 'Backend Team');

// Add a member (must already be an org member)
$teamManager->addMember($team, $user);

// Remove a member
$teamManager->removeMember($team, $user);
```

For team-level resource access, the resource entity must implement `TeamAccessibleInterface`. The `PermissionChecker` cascade automatically checks team access between direct resource access and org-level access.

Invitations

Invite by email

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

$generated = $invitationManager->invite(
    organization: $org,
    email: 'bob@example.com',
    role: $memberRole,
    invitedBy: $currentUser   // optional, for audit trail
);

// $generated->plaintext — build accept URL and email it
// $generated->invitation — the persisted entity
```

Emits `InvitationCreated` with the plaintext token. Subscribe to send the invitation email.

Accept an invitation

```
// GET /invitations/accept?token=abc123...
$membership = $invitationManager->accept($token, $authenticatedUser);
```

Validates:

1. Token is known and active (not expired, not revoked, not accepted).
2. `$authenticatedUser->getEmail()` matches `$invitation->getEmail()` (case-insensitive). This prevents session hijacking — someone logged in with a different email cannot consume another person's invitation.

On success: marks the invitation as accepted, creates the membership via `OrganizationManager::addMember()`, emits `InvitationAccepted`.

Revoke an invitation

```
$invitationManager->revoke($invitation);
```

Only active invitations can be revoked. Emits `InvitationRevoked`.

Purge expired invitations

```
$invitationManager->purgeExpired();
```

Deletes pending-then-expired invitations. Accepted and revoked invitations are kept as audit trail.

Organization context

For request-scoped “which org am I operating in?” resolution:

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

// In a kernel.request listener or middleware:
$orgId = $request->headers->get('X-Organization-Id');
$org = $orgRepository->find($orgId);
$organizationContext->setCurrent($org);

// Anywhere downstream:
$org = $organizationContext->getCurrent();
```

The default implementation (`InMemoryOrganizationContext`) is a stateful singleton. In PHP-FPM this works naturally (container rebuilt per request). For long-running workers (Swoole, RoadRunner), reset the context between requests.

TTL configuration

```
derafu_platform:
    identity:
        invitation_ttl: 604800  # 7 days (default)
```

Last updated on 09/09/2026