Integration with app resources

Traits and interfaces to plug app entities into identity, RBAC, and tenancy.

Anonymous · 09/09/2026

Integration with app resources

The bundle ends where business logic begins. Entities like Project, Invoice, or Document belong to the app — but they often need to participate in the identity model (belong to an org, have collaborators with roles). The bundle provides **interfaces and traits** for this.

OrganizationOwnedInterface + Trait

“This entity belongs to an organization.”

```
use Derafu\PlatformBundle\Identity\Contract\Integration\OrganizationOwnedInterface;
use Derafu\PlatformBundle\Identity\Entity\Trait\OrganizationOwnedTrait;

#[ORM\Entity]
#[ORM\Table(name: 'projects')]
class Project implements OrganizationOwnedInterface
{
    use OrganizationOwnedTrait;

    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(length: 255)]
    private string $name;

    // ... app-specific fields
}
```

The trait adds a `ManyToOne` to `OrganizationInterface` (resolved to your concrete Organization at runtime). The field is **nullable** by default — an entity can exist without an organization (personal account pattern). Add `nullable: false` if org ownership is mandatory.

The `ResourcePermissionVoter` uses this interface for cascading: if a user doesn’t have direct resource-level access, the voter automatically checks their org-level permission on the resource’s owning organization.

UserOwnedInterface + Trait

“This entity belongs directly to a user.”

```
use Derafu\PlatformBundle\Identity\Contract\Integration\UserOwnedInterface;
use Derafu\PlatformBundle\Identity\Entity\Trait\UserOwnedTrait;

class Project implements OrganizationOwnedInterface, UserOwnedInterface
{
    use OrganizationOwnedTrait;
    use UserOwnedTrait;
    // ...
}
```

Both traits are composable. A single entity can be owned by a user OR by an organization (or both, with app logic deciding which applies).

The `IdentityTenantFilter` uses `UserOwnedInterface` to automatically append `WHERE owner_id = :id` to queries when a user context is active.

ResourceAccessibleInterface + BaseResourceAccess

“This entity supports per-resource collaborators with roles.”

Step 1: Create the access pivot entity

```
use Derafu\PlatformBundle\Identity\Entity\BaseResourceAccess;

#[ORM\Entity]
#[ORM\Table(name: 'project_access')]
#[ORM\UniqueConstraint(columns: ['user_id', 'project_id'])]
class ProjectAccess extends BaseResourceAccess
{
    #[ORM\ManyToOne(targetEntity: Project::class, inversedBy: 'accesses')]
    #[ORM\JoinColumn(name: 'project_id', nullable: false, onDelete: 'CASCADE')]
    private Project $project;

    public function __construct(UserInterface $user, RoleInterface $role, Project
$project)
    {
        parent::__construct($user, $role);
        $this->project = $project;
    }

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

Each resource type gets its own table — real FK integrity, no polymorphic hacks.

Step 2: Implement the interface on the resource

```
use Derafu\PlatformBundle\Identity\Contract\Integration\ResourceAccessibleInterface;

class Project implements OrganizationOwnedInterface, ResourceAccessibleInterface
{
    use OrganizationOwnedTrait;

    #[ORM\OneToMany(mappedBy: 'project', targetEntity: ProjectAccess::class,
                    cascade: ['persist', 'remove'], orphanRemoval: true)]
    private Collection $accesses;

    public function __construct()
    {
        $this->accesses = new ArrayCollection();
    }

    public function getResourceAccesses(): Collection
    {
        return $this->accesses;
    }
}
```

Step 3: Use it

```
// In a controller:
#[IsGranted('write', subject: 'project')]
public function edit(Project $project): Response { ... }
```

The `ResourcePermissionVoter` kicks in, iterates the Project's access entries for the current user, and checks permissions. If no direct access is found, it cascades to org-level and then global.

TeamAccessibleInterface

"This entity's access can be granted to teams."

Implement `TeamAccessibleInterface` and add a `BaseTeamResourceAccess` pivot per resource type (similar to `BaseResourceAccess`). The `PermissionChecker` cascade checks team memberships between direct resource access and org-level.

Summary

Interface	Trait	What it adds
`OrganizationOwnedInterface`	`OrganizationOwnedTrait`	`$organization` ManyToOne
`UserOwnedInterface`	`UserOwnedTrait`	`$owner` ManyToOne
`ResourceAccessibleInterface`	(manual collection)	Per-resource RBAC
`TeamAccessibleInterface`	(manual pivot entity)	Team-level RBAC
`BaseResourceAccess`	—	Access pivot MappedSuperclass
`BaseTeamResourceAccess`	—	Team access pivot MappedSuperclass

All integration interfaces live under `Derafu\PlatformBundle\Identity\Contract\Integration\`.

Last updated on 09/09/2026