

# Guide d'Intégration API Auth Service

TraMaSys Engineering

26 novembre 2025

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Authentification Sécurité</b>	<b>2</b>
2.1	Obtenir un Token (Login) . . . . .	2
2.2	Utiliser le Token (Bearer Auth) . . . . .	2
<b>3</b>	<b>Gestion des Droits (RBAC)</b>	<b>4</b>
3.1	Nomenclature des Permissions . . . . .	4
3.2	Cycle de Vie et Dépendances . . . . .	4
3.2.1	Étape 1 : Créer une Permission . . . . .	4
3.2.2	Étape 2 : Créer un Rôle . . . . .	5
3.2.3	Étape 3 : Assigner la Permission au Rôle . . . . .	5
<b>4</b>	<b>Gestion des Utilisateurs</b>	<b>6</b>
4.1	Attribution de Rôles . . . . .	6
4.2	Mise à jour du Profil . . . . .	6
4.3	Sécurité : Changement de Mot de Passe . . . . .	6
<b>5</b>	<b>Conclusion</b>	<b>7</b>

# 1 Introduction

Ce document détaille les procédures d'intégration avec le microservice d'authentification **TraMaSys Auth**. Ce service gère l'identité des utilisateurs, la sécurité via JWT et le contrôle d'accès basé sur les rôles (RBAC).

L'API suit les standards REST et expose une documentation interactive Swagger à l'adresse :

`http://votre-domaine:8080/swagger-ui.html`

## 2 Authentification Sécurité

L'API est entièrement *Stateless*. Aucune session n'est conservée côté serveur. Chaque requête vers une route protégée doit inclure un jeton d'accès (Access Token).

### 2.1 Obtenir un Token (Login)

Pour commencer, l'utilisateur doit s'authentifier avec ses identifiants (Username/E-mail et Mot de passe).

Requête HTTP

**POST** /api/auth/login

Listing 1 – Exemple de Body de requête

```
1 {
2   "identifiant": "brayan_dev",
3   "password": "MonSuperMotDePasse"
4 }
```

Listing 2 – Exemple de Réponse (Succès 200)

```
1 {
2   "accessToken": "eyJhbGciOiJIUzI1NiJ9...",
3   "refreshToken": "eyJhbGciOiJIUzI1NiJ9...",
4   "user": {
5     "id": "a1b2c3d4-...",
6     "username": "brayan_dev",
7     "roles": ["ADMIN"],
8     "permissions": ["auth:user:delete", "stock:item:read"]
9   }
10 }
```

### 2.2 Utiliser le Token (Bearer Auth)

Une fois l'`accessToken` obtenu, il doit être envoyé dans les en-têtes (Headers) de **toutes** les requêtes subséquentes.

### Attention

Le format de l'en-tête doit respecter scrupuleusement la syntaxe suivante :

- **Header Name** : Authorization
- **Header Value** : Bearer <votreToken>

*Exemple d'en-tête HTTP brut :*

Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWliOiIxMjM0NTY...

Si ce header est manquant ou mal formé, l'API retournera une erreur **401 Unauthorized**.

## 3 Gestion des Droits (RBAC)

Le système repose sur un contrôle d'accès basé sur les rôles (RBAC). Cependant, la granularité fine est gérée par des **Permissions**.

Un utilisateur possède des Rôles (ex : MANAGER), et ces Rôles possèdent des Permissions (ex : stock:produit:create).

### 3.1 Nomenclature des Permissions

Pour garantir une cohérence entre tous les services de l'écosystème TraMaSys, les permissions **doivent** respecter une convention de nommage stricte.

#### Format Obligatoire

[service] : [Objet] : [action]

- **[service]** : Nom de l'application ou du microservice concerné (ex : auth, syndicat, rideAndGo).
- **[Objet]** : La ressource manipulée (ex : user, product, invoice).
- **[action]** : L'action effectuée (ex : create, read, update, delete, approve).

#### Exemples valides :

- auth:user:delete (Droit de supprimer un utilisateur dans le service Auth)
- stock:inventory:read (Droit de voir l'inventaire dans le service Stock)
- rh:contract:sign (Droit de signer un contrat dans le service RH)

### 3.2 Cycle de Vie et Dépendances

L'attribution des droits suit une séquence logique stricte. Vous ne pouvez pas lier des éléments qui n'existent pas encore en base de données.

1. **Création de la Permission** : Définir l'action atomique (respectant la nomenclature).
2. **Création du Rôle** : Définir le profil métier (ex : ADMIN, DRIVER).
3. **Association** : Lier la Permission au Rôle.
4. **Affectation** : Donner le Rôle à un Utilisateur.

#### 3.2.1 Étape 1 : Créer une Permission

POST /api/permissions

Il est impératif de créer la permission avant de pouvoir l'utiliser.

```
1 {  
2   "name": "rideAndGo:vehicule:create",  
3   "description": "Permet de creer une nouveau vehicule"  
4 }
```

### 3.2.2 Étape 2 : Créer un Rôle

POST /api/roles

Créez le conteneur logique qui regroupera plusieurs permissions.

```
1 {  
2   "name": "MAGASINIER"  
3 }
```

### 3.2.3 Étape 3 : Assigner la Permission au Rôle

POST /api/permissions/assign/{roleName}

Lie la permission créée à l'étape 1 au rôle créé à l'étape 2.

```
1 // POST /api/permissions/assign/MAGASINIER  
2 {  
3   "permissionName": "rideAndGo:vehicule:create"  
4 }
```

## 4 Gestion des Utilisateurs

Une fois les Rôles et Permissions configurés, l'étape finale consiste à gérer les utilisateurs et leurs accès.

### 4.1 Attribution de Rôles

L'utilisateur n'hérite pas directement de permissions. Il hérite de **Rôles**, qui eux contiennent les permissions. C'est ce mécanisme qui permet de charger automatiquement les droits dans l'attribut `permissions` de l'objet User lors de la connexion.

**POST** `/api/users/{id}/roles/{roleName}`

Ajoute un rôle existant à un utilisateur.

*Exemple : Promouvoir l'utilisateur "brayan" (ID 123...) en ADMIN.*

**Paramètres d'URL :**

- `id` : UUID de l'utilisateur.
- `roleName` : Nom du rôle (ex : ADMIN).

*Aucun corps (Body) n'est requis pour cette requête.*

**DELETE** `/api/users/{id}/roles/{roleName}`

Retire un rôle à un utilisateur (Révocation d'accès).

### 4.2 Mise à jour du Profil

Permet de modifier les informations personnelles sans toucher aux données sensibles (mot de passe, rôles).

**PUT** `/api/users/{id}`

Mise à jour des informations de base.

```
1 {  
2   "firstName": "Brayan",  
3   "lastName": "Armel",  
4   "phone": "+237600000000"  
5 }
```

### 4.3 Sécurité : Changement de Mot de Passe

Pour des raisons de sécurité, le changement de mot de passe nécessite de fournir l'ancien mot de passe pour validation.

**PUT** `/api/users/{id}/password`

Route sécurisée pour la rotation de mot de passe.

```
1 {  
2   "currentPassword": "AncienPassword123!",  
3   "newPassword": "NouveauPasswordSecure456!"  
4 }
```

## 5 Conclusion

Ce service d'authentification est conçu pour être la pierre angulaire de la sécurité de l'écosystème TraMaSys.

### **Rappel des bonnes pratiques :**

1. Ne jamais stocker les mots de passe en clair (le service les hache automatiquement).
2. Toujours passer par l'étape de création de permission ([service]:[objet]:[action]) avant de tenter une assignation.
3. Utiliser le `refreshToken` pour renouveler les sessions sans demander à l'utilisateur de se reconnecter.

Pour toute assistance technique, veuillez vous référer à l'équipe Backend.