# Terminologies

|  |  |
| --- | --- |
| Mot | Définition |
| Ressource | Une ressource est un objet qui doit être protégé. [[1]](#footnote-1) |
| Dossier | Ressource qui peut en contenir d’autres |
| Fichier | Ressource unique. |
| Politique d’autorisation | Associée à une ou plusieurs ressources et est constituée d’une ou plusieurs règles. Elle est utilisée durant la prise de décision du serveur d’autorisation d’accorder ou non des permissions à un client sur une ressource (RPT token) [[2]](#footnote-2) |
| Règle d’autorisation | Appartient à une politique |
| Resource owner | Une entité capable d’accorder l’accès à une ressource protégée. Lorsqu’un resource owner est une personne on parle alors d’utilisateur final [[3]](#footnote-3). |
| Scope | Représente une liste de ressources pouvant être accédées par un client. |
| Client | Une application qui exécute des requêtes sur une ressource protégée avec le consentement du resource owner. |
| Serveur UMA | Serveur qui respecte l’implémentation du RFC-OPENID [[4]](#footnote-4) |
| Serveur OpenId | Serveur qui respecte l’implémentation du RFC UMA [[5]](#footnote-5) |
| RPT token | Token spécifique à UMA utilisé pour vérifier l’autorisation. |

# Cas d’utilisations du manager

## Protéger les ressources

<<include>>

<<include>>

<<include>>

<<include>>

<<include>>

Créer un dossier

Créer un fichier

Déplacer les ressources

Créer une ressource

Dupliquer une ou plusieurs ressources

Afficher le moyen d’accéder à une ressource

Renommer une ressource

<<include>>

Supprimer une ou plusieurs ressources

Déplacer une ressource vers un sous répertoire

Administrateur

Télécharger une ou plusieurs ressources

Afficher les raccourcis clavier

Rechercher une ressource dans le répertoire de travail

Changer de visualisation

Rafraîchir le répertoire courant

Administrateur

Naviguer vers le répertoire suivant / précédent

Ajouter / Editer les règles d’autorisation

Sélectionner toutes les ressources

### Action : Déplacer les ressources

Créer deux répertoires « Folder1 » et « Folder2 ». Ajouter un fichier « file.txt » dans le répertoire « Folder1 » puis ajouter une règle d’autorisation à la ressource « Folder1 ».

Vérifiez dans la table « dbo.PolicyResource » que deux records ont été insérés.

|  |  |
| --- | --- |
| Tests | Nombre enregistrements dans « dbo.PolicyResource » |
| Déplacez le fichier « file.txt » vers le répertoire « Folder2 » | 1 |
| Déplacez le fichier « file.txt » vers le répertoire « Folder1 » | 2 |
| Ajoutez un nouveau fichier « file2.txt » dans le répertoire «Folder1 » et déplacez « file.txt » & « file2.txt » vers le répertoire « Folder2 » | 1 |
| Re-exécutez l’étape précédente mais en déplaçant cette fois les fichiers « file.txt » & « file2.txt » vers le répertoire « Folder1 » | 3 |
| Ajoutez un nouveau sous répertoire « SubFolder1.1 » à la ressource « Folder1 » puis ajoutez lui une nouvelle règle d’autorisation | 5 |
| Déplacez les deux fichiers « file.txt » et « file2.txt » vers ce nouveau sous répertoire | 7 |

### Action : Dupliquer une ou plusieurs ressources

Sélectionnez le fichier « file.txt » et dupliquez le en exécutant l’une des actions :

* Dans le menu contextuel cliquez sur *copy* puis sur *paste* ou directement sur *duplicate*.
* Utilisez les raccourcis clavier « CTRL+C & CTRL+V ».



La table « dbo.PolicyResource » devrait contenir 9 enregistrements et un nouveau fichier devrait apparaître dans le répertoire « SubFolder1.1 ».

### Action : Télécharger une ou plusieurs ressources

Il existe plusieurs façons de télécharger des ressources :

* Dans votre l’explorateur Windows, sélectionnez un ou plusieurs fichiers et déplacez les vers la fenêtre « Protect your resources » du site
* Dans le menu contextuel du répertoire de travail, cliquez sur l’option *upload files*.



Vérifiez que les points suivants sont corrects :

* Une ou plusieurs ressources doivent être créées
* Les politiques d’autorisation doivent être mises à jour.

### Action : Afficher le moyen d’accéder à une ressource

Sélectionnez n’importe quelle ressource et affichez son menu contextuel. Cliquez sur l’option : *how to access ?* , une nouvelle fenêtre s’affiche. Vérifiez si l’URL de la ressource est correcte et commence bien par « resources ».



### Action : Renommer une ressource

Sélectionnez une ressource et choisissez parmi l’une des options :

* Raccourci clavier « F2 »
* Action *Rename*  du menu contextuel
* Icône 

### Action : Supprimer une ou plusieurs ressources

Sélectionnez une ressource et choisissez l’une des options :

* Raccourci clavier « DEL »
* Action *Delete* du menu contextuel
* Icône 

### Action : Déplacer une ressource vers un sous-répertoire

Sélectionnez une ressource et dans son menu contextuel cliquez sur *into new folder* . La ressource sera déplacée dans un nouveau sous répertoire.



### Action : Afficher les raccourcis clavier

La liste des raccourcis clavier est accessible par le raccourci *F1* :



### Action : Rechercher une ressource dans le répertoire de travail

Accessible par le raccourci clavier *CTRL+F* ou la bar de rechercher en haut à droite de l’écran.



### Action : Changer de visualisation

Pour basculer entre la visualisation verticale et horizontale des ressources cliquez sur cet icône : 

### Action : Rafraîchir le répertoire courant

Action accessible soit par le raccourci clavier *F5* ou le menu contextuel *Reload*



### Action : Créer un nouveau répertoire

Dans votre répertoire de travail sélectionnez l’une des options :

* Action *New folder* du menu contextuel
* Icône 

### Action : Créer un nouveau fichier

Dans votre répertoire de travail sélectionnez l’une des options :

* Action *New file* du menu contextuel
* Icône 

### Action : Naviguer vers le répertoire précédent / suivant

L’historique de navigation est sauvegardé dans votre navigateur et il est possible de naviguer vers le répertoire précédent ou suivant.

* Utilisez les boutons  situés dans la bar de recherche
* Raccourcis claviers *Back* ou *CTRL+←* pour naviguer vers le répertoire précédent.
* Raccourci clavier *CTRL+→* pour naviguer vers le répertoire suivant.

### Action : Ajouter / Editer les règles d’autorisation

Lorsqu’une ressource est sélectionnée il est possible d’ajouter ou éditer ses règles d’autorisation. Cliquez sur l’icône  ou le raccourci clavier *CTRL+P*.

Dans la nouvelle fenêtre affichée au premier plan, il est possible d’éditer les règles de votre politique d’autorisation. Ces règles sont exclusives c’est-à-dire que si l’une d’entre elles est correcte alors l’autorisation est accordée.

Une politique d’autorisation est composée d’au moins une ou plusieurs règles. Une règle accorde des permissions d’accès à une ressource pour des clients et des valeurs de claims donnés. Il est difficile pour une personne n’ayant pas de connaissance en UMA et OPENID de définir ces règles. Le scénario suivant décrit la démarche à suivre afin d’identifier les règles d’autorisation en tentant de résoudre un problème d’autorisation.

Problématique : Seul l’utilisateur thabart de l’application Sample Client est habilité à consulter son compte bancaire N°12345.

Solution : La première étape consiste à identifier les identités et de les classifier selon leur nature. Ici quatre informations importantes ressortent (soulignées) du scénario. Elles ont été classifiées grâce au tableau de décision suivant :

|  |  |
| --- | --- |
| Question | Nature |
| Quel est l’identifiant de la ressource à protéger ? | Identifiant de la ressource |
| Quel est l’identifiant de votre client ? | Client Id |
| Quelles actions souhaitez vous exécuter sur cette ressource ? | Permissions |
| Souhaitez vous restreindre l’accès à un ou plusieurs utilisateurs ? Si oui quelles informations peuvent être utilisées pour les identifier ? | Claims |
| Pouvez-vous identifier la nature de votre ressource ? | Nature |

Voici les informations classifiées :

* Identifiant de la ressource : N°12345
* Client Id : Sample Client
* Permissions : consulter
* Claims : thabart
* Nature : BankAccount

La seconde étape consiste à créer la ressource. Si vous avez réussi à identifier la nature de votre ressource vous pouvez alors créer un répertoire BankAccount.

Naviguez ensuite vers ce nouveau répertoire et créez un fichier avec comme nom l’identifiant de la ressource N°12345.



Enfin ouvrez l’éditeur des règles d’autorisation et ajouter une nouvelle règle. Sous l’intitulé *allowed clients* sélectionnez le client correspondant au *Client Id* (Sample Client). Sous *allowed claims* sélectionnez *sub* et rentrez la valeur de votre claim (thabart) puis cliquez sur *Add* afin d’ajouter le claim dans la liste.

Cochez les permissions qui ont été identifiés (read).

Ajoutez la nouvelle règle en cliquant sur *Add rule* puis sur *save* pour sauvegarder.



Une ressource qui possède une politique d’autorisation doit être encadrée par un cadre rouge.

### Action : Sélectionner toutes les ressources

Toutes les ressources du répertoire de travail peuvent être sélectionnées avec le raccourci *CTRL+A*.

Vous avez sûrement dû remarquer que les actions disponibles dans les *assets* n’ont pas été décrites, cela fera l’objet de la prochaine partie.

## Editer les actifs (openid & uma)

<<include>>

<<include>>

Supprimer un ou plusieurs clients

Ajouter un client

Administrateur

Editer un client

Afficher les informations d’un scope

Supprimer un ou plusieurs scopes

Supprimer un ou plusieurs resource owners

Editer un resource owner

Supprimer une ou plusieurs politiques d’autorisation

Afficher les informations d’une politique d’autorisation

Supprimer une ou plusieurs ressources

Afficher les informations d’une ressource

### Action : Ajouter un client

Ouvrez le répertoire *assets>>openid>>clients* puis affichez son menu contextuel. Cliquez sur *add client* et rentrez une ou plusieurs urls de redirection séparées par une virgule.

Attention : Les urls doivent obligatoirement commencées par https et ne peuvent pas avoir comme nom d’hôte localhost [[6]](#footnote-6)

### Action : Supprimer un ou plusieurs clients

Sélectionnez un ou plusieurs clients puis cliquez sur  dans le menu contextuel.

### Action : Editer un client

Affichez les informations d’un client en cliquant sur  dans le menu contextuel.



L’identifiant du client, son secret ainsi que ses urls de redirection sont affichées. Pour éditer ses propriétés cliquez sur *Edit*.



Les paramètres sont expliqués dans le RFC OpenId [[7]](#footnote-7).

De façon générale vous n’aurez pas besoin d’éditer les paramètres avancées sauf si vous souhaitez changer le workflow. Voici quelques scénarios où vous devez envisager de les modifier.

* Vous disposez d’une URL JWKS qui peut être utilisée pour décrypter et vérifier la signature du paramètre *request [[8]](#footnote-8)*.
* Modifier la façon dont le client va s’authentifier au point d’entrée token.
* Les informations affichées dans la vue *consent*.

Si vous souhaitez avoir plus d’information sur les paramètres je vous invite à nous contacter ou si vous en avez le courage de lire la documentation openid [[9]](#footnote-9).

### Action : Supprimer un ou plusieurs scopes

Ouvrez le répertoire *assets>>openid>>scopes*, sélectionnez un ou plusieurs scopes puis cliquez sur  dans le menu contextuel.

### Action : Afficher les informations d’un scope

Sélectionnez un scope et cliquez sur  dans le menu contextuel. Des informations relatives au scope sont affichées comme :

* Open id scope : Indique si c’est un scope OpenId
* Displayed in the consent screen : Indique si le scope est affiché sur l’écran consent.
* Exposed in the contract : indique si le scope est retourné par le endpoint /.well-known/openid-configuration.

### Action : Supprimer un ou plusieurs resource owners

Sélectionnez un ou plusieurs resource owners puis cliquez sur  dans le menu contextuel.

### Action : Editer un resource owner

Pour ajouter des rôles à un resource owner il doit être déclaré comme un compte local, autrement le message « not a local account » est affiché.

Pourquoi avoir besoin d’ajouter des rôles ? Parfois nous ne pouvons pas nous baser uniquement sur des claims provenant des fournisseurs d’identité externes. Le claim role n’est pas standard à OpenId mais il l’est pour un fournisseur de token SAML comme ADFS. Lorsque vous éditez la politique d’autorisation d’accès à une ressource, il est parfois utile de filtrer sur les rôles des resource owners et non sur des données personnelles comme le nom.

Comme il a été mentionné plus haut le resource owner peut avoir deux états externe ou interne.

Externe

Interne

Utilisateur créé un compte local

Utilisateur confirme son enregistrement comme compte local

Utilisateur s’authentifie avec un fournisseur d’identité externe

Lorsqu’un utilisateur s’authentifie avec un fournisseur d’identité externe comme « Twitter », « Hotmail » ou encore « GitHub » alors un compte est automatiquement créé avec un état « externe ».



Pour basculer vers un compte local, l’utilisateur doit cliquer sur « create a local account » :



Maintenant que vous avez un resource owner local, ouvrez son menu contextuel et cliquez sur . Les rôles peuvent être ajoutées dans la nouvelle fenêtre affichée au premier plan, persistez les changements en cliquant sur *update*.



### Action : Supprimer une ou plusieurs politiques d’autorisation

Ouvrez le répertoire *assets>>uma>>authorization policies* puis sélectionnez une ou plusieurs politiques puis cliquez sur  dans le menu contextuel.

### Action : Afficher les informations d’une politique d’autorisation

Sélectionnez une politique puis cliquez sur  dans le menu contextuel. La fenêtre affichée au premier plan affiche la liste des ressources impactées par la politique et le nombre de règles.



### Action : Supprimer une ou plusieurs ressources

Ouvrez le répertoire assets>>uma>>resources puis sélectionnez une ou plusieurs ressources puis cliquez sur  dans le menu contextuel.

### Action : Afficher les information d’une ressource

Sélectionnez une ressource et cliquez sur  dans le menu contextuel. La fenêtre affiche le nom de la ressource et ses scopes



## Fournisseurs d’identité

Le serveur d’autorisation offre la possibilité à l’utilisateur final de pouvoir s’authentifier avec l’un des fournisseurs d’identité externe au choix tels que : Hotmail, ADFS, GitHub, ou encore votre carte d’identité belge.

Lors de la vérification de la politique d’autorisation, ce sont les claims provenant de ces fournisseurs externes qui sont utilisés. N’oubliez pas qu’il est toujours possible d’en ajouter (référez vous à la description de l’action : « *Editer un resource owner* »).

Voici les actions disponibles sur l’écran « Connections » :

Activer / désactiver un fournisseur

Editer les paramètres d’un fournisseur

Ajouter un fournisseur

Lister les fournisseurs

Les fournisseurs d’identité externes suivants sont configurés par défaut : Microsoft, Linkedin, Google, GitHub, Facebook ainsi que la carte d’identité belge.

### Action : Lister les fournisseurs

Cliquez sur l’onglet « Connections » du site. Une fenêtre avec la liste des fournisseurs d’identité externes s’affiche.



### Action : Activer / désactiver un fournisseur

Les fournisseurs peuvent être activés / désactivés de la page d’authentification. Pour cela rien de plus simple il suffit de cliquer sur le bouton « On / Off » situé à côté de chaque fournisseur.

Par défaut voici à quoi ressemble la page d’authentification :



Voici la nouvelle page lorsque « Facebook » est désactivé :



L’administrateur peut rafraîchir la liste des fournisseurs sans se soucier de relancer l’application.

### Action : Ajouter un fournisseur

Pour ajouter un fournisseur cliquez sur le bouton « New identity provider ». Une nouvelle fenêtre s’affiche au premier plan. Remplissez les champs en vous aidant du tableau explicatif ci-dessous. Lorsque vous avez terminé, cliquez sur le bouton « Create » afin de confirmer la création.

|  |  |
| --- | --- |
| Nom champ | Description |
| Name | Nom qui sera affiché sur la page d’authentification |
| Callback path | Donnée technique utilisée par le serveur d’autorisation. Faîtes bien attention à respecter la convention de nommage « /signin-\* » autrement le bouton ne sera pas affiché |
| Choose an identity provider type | Choisissez le type de fournisseur d’identité. |

Si vous ne trouvez pas le bon type de fournisseur vous pouvez nous soumettre un nouveau ticket « enhancement» sur notre site. La fonctionnalité sera développée dans les plus brefs délais et délivrée dans une nouvelle image Docker ou par XCOPY des dlls.

Dans le cas où vous souhaitez utiliser votre base de données des utilisateurs. On peut vous aider à migrer vos données vers la nouvelle table ou créer un fournisseur d’identité OpenId qui récupère les informations utilisateur de votre base de données.

### Action : Editer les paramètres d’un fournisseur

Pour éditer un fournisseur cliquez sur son titre. Les paramètres qui sont affichés sont différents selon le type de fournisseur. Le tableau qui suit explique ces différences.

|  |  |  |
| --- | --- | --- |
| Type | Paramètres | Options |
| OAUTH2.0 | Callback path | TokenEndpoint |
| Namespace | AuthorizationEndpoint |
| Class | Scope |
| Code | ClientId |
| UserInformationEndpoint |
| ClientSecret |
| OPENID | Callback path | ClientId |
| Scope |
| ClientSecret |
| WellKnownConfigurationEndPoint |
| WSFEDERATION | Callback path | Realm |
| Namespace | IdEndpoint |
| Class |
| Code |

Vous aurez noté quelques différences dans les paramètres selon le type de fournisseur. Namespace, Code et Class n’existent pas dans OpenId car nous nous assumons ne pas avoir besoin d’un parseur de Claims si le fournisseur d’identité respecte bien le RFC OpenId.

Le code « OAUTH2.0 » doit respecter cette signature :

*public List<Claim> Process(JObject jObj)*

Tandis que le code « WS-Federation » doit respecter celle-ci :

*public List<Claim> Process(XmlNode node)*

Le namespace et le nom de la classe doivent être renseignées dans les champs qui leurs sont dédiés. Si par exemple le code du parseur est le suivant alors le namespace est « Parser » et le nom de la classe est « Example ».

*namespace Parser*

*{*

*public class Example*

*{*

*public List<Claim> Process(XmlNode node) { }*

*}*

*}*

La liste des Options diffèrent selon le type cela se justifie par une différence d’implémentation décrite dans les RFCs. Pour plus d’informations lisez les RFCs OpenId [[10]](#footnote-10), OAUTH2.0[[11]](#footnote-11) et WS-Federation [[12]](#footnote-12).

## Paramètres

Affichez les paramètres en cliquant sur l’onglet « Settings ». Les durées de validité en secondes des tokens (access token ou identity token) et code d’autorisation peuvent être mises à jour.



# Cas d’utilisations de l’extension Visual Studio

L’extension est disponible que pour la version 2015 de Visual Studio. D’autres versions antérieurs seront supportées à l’avenir.

## Configurer les paramètres

Il est possible de configurer l’URL utilisé afin d’ajouter les ressources. Ouvrez les options de « SimpleIdentityServer » en cliquant sur « Tools > Options > SimpleIdentityServer » puis mettez à jour l’URL.



## Générer une ou plusieurs ressources

Pour ouvrir la fenêtre sélectionnez le projet API qui contient les ressources, puis cliquez sur l’option  dans le menu contextuel.

N’ayez pas peur si la liste des opérations prend du temps à se charger.

Si après un certain temps aucune opération n’est affichée alors fermez la fenêtre, re-sélectionnez le projet API et rouvrez la fenêtre.

Rafraîchir la liste des ressources

Protéger une ou plusieurs ressources

Développeur

### Rafraîchir la liste des ressources

Pour rafraîchir la liste des ressources disponibles, cliquez sur le bouton « refresh » et attendez quelques secondes avant que la liste s’affiche.



### Protéger une ou plusieurs ressources

Sélectionnez une ou plusieurs ressources que vous souhaitez protéger et indiquez le numéro de version de votre API. Ici, La notion de version est très importante car si des changements cassants ont été apportés à l’API, il suffit d’incrémenter le numéro et déployer la nouvelle version sans toucher à l’ancienne. Cliquez ensuite du « Protect » pour protéger les ressources sélectionnées.

L’opération va effectuer plusieurs tâches :

* Créer une ressource en suivant une convention de nommage :: « *Apis\<nom de l’assembly>\<numéro de version>\<nom du contrôleur>\<action> »*
* Ajouter les Nuget packages « SimpleIdentityServer.UmaIntrospection.Authentication » et « SimpleIdentityServer.Uma.Authorization »
* Ajouter un fichier de test « Startup\_Sample.cs » que vous pourrez réutiliser afin de protéger vos opérations.



Les détails techniques sont expliqués dans le chapitre « Documentation technique ».

## Générer un proxy de sécurité

Cette fonctionnalité est utilisée par le client afin de récupérer le RPT token. Ce dernier est passé en paramètre des requêtes qui souhaitent accéder à une ressource protégée. Pour afficher la fenêtre sélectionnez le projet du client et cliquez sur l’option « generate security proxy » dans le menu contextuel.

Rechercher une ou plusieurs ressources

Ajouter le proxy de sécurité

Développeur

### Action : Rechercher une ou plusieurs ressources

Tapez le nom de la ressource dans la bar de recherche et cliquez sur « Search ».



### Action : Ajouter le proxy de sécurité

Sélectionnez une ressource puis cliquez sur « Add ». Les tâches suivantes seront exécutées :

* Installer le Nuget package : « SimpleIdentityServer.Proxy »
* Ajouter deux fichiers au projet :
  + SecurityProxy\_\*.cs : possède une méthode statique qui retourne le RPT token.
  + AuthProvider.cs : possède une méthode statique qui retourne l’identity token.

# Scénarios

Cette partie décrit les problématiques de sécurité les plus fréquentes que l’on retrouve en entreprise et explique comment les résoudre.

Le code source se trouve ici : *https://github.com/thabart/SimpleIdentityServer.Samples.git*

## Premier scénario : Application lourde souhaite accéder à une opération protégée

**Contexte** : une entreprise d’e-commerce a développé en interne une application lourde. Elle est utilisée par son équipe de Marketing afin de récupérer des informations sur ses clients les plus fidèles. L’application a été développée en WPF et interroge un service web afin de récupérer la liste des clients. Seule cette application et les utilisateurs qui appartiennent au groupe «marketing » sont autorisés à récupérer la liste.

**Problème** : Comment l’application WPF peut accéder à l’opération protégée ?

**Solution** :

Le «workflow » que l’on souhaite implémenter est décrit dans le graphe ci-dessous. Il se compose de trois grandes étapes :

* *Identity token* : Récupérer un identity token avec le grant-type implicit [[13]](#footnote-13). Le token est ensuite retourné au client en tant que paramètre du callback.
* *RPT token* : Le client (application WPF) récupère un RPT Token [[14]](#footnote-14) pour cela il s’authentifie auprès du serveur UMA avec ses identifiants et passe l’identity token du resource owner. Ces informations sont requises par la politique d’autorisation, pour prendre la décision d’accorder ou non l’autorisation au client d’accéder à la ressource protégée. Pour plus d’informations concernant la politique d’autorisation référez à la description du cas d’utilisation : « Ajouter / Editer les règles d’autorisation ».
* *Vérification du RPT token*: Le RPT token est passé dans le header « Authorization » de la requête exécutée pour récupérer la liste des clients. Lorsqu’elle est reçue par l’API, le token est récupéré et vérifié en utilisant le point d’introspection [[15]](#footnote-15) offert par le serveur UMA.

Application WPF

Service API

Serveur OPENID

Récupérer Identity Token

Identity token

Serveur UMA

Authentifier utilisateur

Récupérer RPT token

RPT token

Récupérer clients

Nous vous épargnons les détails d’implémentation afin de faciliter la compréhension. Le workflow présenté est normalement plus complexe et possède beaucoup plus d’étapes intermédiaires.

Avant d’implémenter la solution, nous allons procéder par étapes de la même façon que le cas d’utilisation expliqué plus haut « ajouter / éditer les règles d’autorisation ».

Identifier et catégoriser les entités

Ajouter un client

Ajouter une ressource

Développer

Ajouter politique d’autorisation

Ajouter le rôle marketing au resource owner

### Identifier et catégoriser les entités

Dans un premier lieu il est nécessaire d’identifier les entités du contexte et de les catégoriser. Pour ce fait vous pouvez utiliser le tableau suivant.

|  |  |
| --- | --- |
| Questions | Type |
| Quelle est l’application consommateur du service que vous souhaitez protéger ? | Client |
| Quelle est l’opération du service que vous souhaitez protéger ? Identifiez le nom du service, sa version, l’entité business (client, produit) et l’opération. | Ressource : concaténation du nom du service, sa version, l’entité business ainsi que l’opération |
| Quelles sont les applications autorisées à accéder à la ressource ? | Liste des clients autorisés |
| Quelles sont les informations des resource owners acceptées pour accéder à la ressource ? | Liste des claims |

Voici ce que l’on obtient après analyse :

* Client : application WPF
* Ressource : ClientApi / v1 / Clients / Get
* Liste des clients autorisés : application WPF
* Liste des claims : rôle marketing.

Maintenant que les entités ont été identifiés vous pouvez utiliser le manager de ressources pour les créer.

### Ajouter un client

Ajoutez un nouveau client et éditez ses propriétés (lisez la description des cas d’utilisation « ajouter un client » & « éditer un client »). Dans la nouvelle fenêtre mettez à jour les propriétés avec les informations que vous avez réussi à extraire. Certains paramètres comme le nom & l’url de redirection sont assez facile à trouver ce qui n’est pas le cas des autres.

Essayez d’abord de deviner les grant\_types dont a besoin le client afin d’interagir avec les différents composants, les autres valeurs pourront ensuite être déduites.

Le workflow présenté ci-dessous requière une seule interaction avec le serveur OpenId. Elle débouche sur l’obtention d’un identity token et d’un access token valide pour les scopes « uma\_protection » et « uma\_authorization » (cet access token sera réutilisé plus tard lors de l’échange du RPT token et cette règle est toujours vraie).

Application WPF

Service API

Serveur OPENID

Récupérer Identity Token

Identity token & access token

Serveur UMA

Authentifier utilisateur

Récupérer RPT token

RPT token

Récupérer clients

Selon la façon dont est implémentée votre application WPF, le grant-type sera différent. Voici les relations entre le type d’implémentation et le grant type.

|  |  |  |  |
| --- | --- | --- | --- |
| Implémentation | Grant-Type | Avantages | Désavantages |
| Navigateur embarqué dans votre application | implicit | L’authentification est délégué au serveur OPENID | Pas le contrôle sur look & feel de la page d’authentification |
| Créer un formulaire | client\_credentials | Contrôle sur le look & feel | Le code de votre application doit être obfusqué pour ne pas exposer le ClientId et Clientsecret.  Cette approche n’est pas conseillé lorsqu’il n’y a pas de relation de confiance entre le client et le serveur. |

Lorsque les grant types ont été identifiés alors les autres paramètres peuvent être déduits. Pour ce fait aidez vous des tableaux ci-dessous :

|  |  |
| --- | --- |
| Grant type | Response types |
| Authorization code | Authorization code |
| Implicit flow | Token |
| Authorization code |
| Identity token |
| Client credentials |  |
| Password |  |
| Refresh token |  |

Relations entre les grant types et response types

|  |  |
| --- | --- |
| Type token | Scopes |
| Récupérer un RPT token | uma\_authorization |
| uma\_protection |
| Récupérer un identity token | Openid |
| Profile |
| Role |
| *La liste complète des scopes se trouve dans la documentation [[16]](#footnote-16)* |

Relations entre le type de token et les scopes

Lorsque vous avez tous les paramètres, la page d’édition peut être complétée de cette façon :

* Callback uris : <https://client.com>
* Grant types : implicit
* Response types : token, id\_token
* Scopes : openid, profile, role, uma\_authorization, uma\_protection



### Ajouter une ressource

La ressource peut être ajoutée de deux façons : avec l’interface graphique (voir les cas d’utilisation : « créer un nouveau répertoire » et « créer un nouveau fichier » ) ou l’extension Visual Studio (voir le cas d’utilisation : « Générer une ou plusieurs ressources »). Dans tous les cas la convention de nommage doit être respectée. Si la ressource est l’opération d’une API alors nous suggérons de respecter cette règle :

*Apis\<nom de votre application>\<numéro de version>\<nom de l’entité métier ou contrôleur>\<nom de l’opération>.*

Dans notre exemple nous obtenons « *Apis\ClientApi\v1\ClientsController\Get* ».

Si vous utilisez l’extension Visual Studio vous ne devez pas vous soucier de cette convention car elle est respectée.

Si la nature de la ressource est autre chose qu’une opération alors essayez de trouver une logique dans votre organisation. Par exemple si la ressource est l’image de profil d’un utilisateur, vous pouvez avoir quelque chose comme : « images\profile\user.png ».

Nous insistons sur l’importance de bien organiser vos ressources, si dès le départ vous partez sur une mauvaise organisation et que plus tard des corrections sont apportées. L’impact sera d’autant plus grand qu’il faudra mettre à jour les URLs dans toutes les applications et redéployer une nouvelle version.

### Ajouter la politique d’autorisation

Maintenant que le client et la ressource ont été créés, ajoutez la politique d’autorisation à la nouvelle ressource. Si vous n’y arrivez pas aidez-vous alors de la méthodologie qui est expliquée dans le chapitre « Action : Ajouter / Editer les règles d’autorisation ».

Voici le résultat attendu :



### Ajouter le rôle marketing au resource owner

Le rôle « marketing » doit être ajouté au resource owner autrement la politique d’autorisation ne passera jamais. Editez les propriétés du resource owner de votre choix et ajoutez lui le rôle « marketing ».



### Développer

Lorsque les entités ont été créées les modifications peuvent être apportées sur l’API et l’application WPF.

#### Modifier l’API

Pour rappel il existe deux types d’autorisation (pour plus de détails référez vous au chapitre « Documentation technique »)  :

* Convention: l’url de la ressource doit correspondre exactement à la structure du projet.
* Individuelle : limiter l’accès aux ressources en indiquant leurs URLs et leurs scopes.

L’implémentation technique marche seulement pour les projets ASP.NET CORE. Si vous travaillez sur une version antérieure, on peut vous aider à écrire votre implémentation. Les nuget packages « *SimpleIdentityServer.UmaIntrospection.Authentication* » et « *SimpleIdentityServer.Uma.Authorization* »  doivent être installées sur votre projet API.

L’autorisation par convention est activée en ajoutant le code suivant dans la procédure « ConfigureServices » de votre classe Startup.

// Authorization policy

services.AddAuthorization(options =>

{

// Add conventional uma authorization

options.AddPolicy("uma", policy =>

{

// policy.Requirements.Add(new ConventionalUmaAuthorizationRequirementTst(null));

policy.AddConventionalUma();

// options.AddPolicy("resourceSet", policy => policy.AddResourceUma("<url>", "<read>","<update>"));

});

});

L’opération « ClientController/Get » doit ensuite être décorée par l’attribut : 

#### Modifier l’application WPF

Avant de commencer n’oubliez pas d’installer le nuget package « SimpleIdentityServer.Proxy ».

Votre application WPF doit posséder une fenêtre d’authentification. Elle peut afficher la fenêtre d’autorisation du serveur OPENID dans un navigateur embarqué ou afficher son propre formulaire. Quelque soit la méthodologie choisie, il est important de stocker l’identity token qui a été récupéré.

Ensuite utilisez le code suivant pour récupérer le RPT token. N’oubliez pas bien évidemment de mettre à jour les propriétés.

public static async Task<string> GetRptToken(

string idToken,

string umaProtectionToken,

string umaAuthorizationToken)

{

var factory = new SecurityProxyFactory();

var proxy = factory.GetProxy(new SecurityOptions

{

ClientId = Constants.ClientInfo.ClientId,

ClientSecret = Constants.ClientInfo.ClientSecret,

UmaConfigurationUrl = "https://localhost:5445/.well-known/uma-configuration",

OpenidConfigurationUrl = "https://localhost:5443/.well-known/openid-configuration",

RootManageApiUrl = "https://localhost:5444/api"

});

try

{

var result = await proxy.GetRpt("resources/Apis/ClientApi/v1/ClientsController/Get", idToken, umaProtectionToken, umaAuthorizationToken, new List<string>

{

"execute"

});

return result;

}

catch (Exception ex)

{

return null;

}

}

Appelez la méthode GetRptToken en n’oubliant pas de passer en paramètre les tokens (identity & access) puis passez le résultat dans le header de votre requête.

var request = new HttpRequestMessage

{

Method = HttpMethod.Get,

RequestUri = new Uri("http://localhost:5100/api/clients")

};

request.Headers.Add("Authorization", $"Bearer {rptToken}");

Si vous voulez exécuter l’exemple complet ouvrez la solution « Scenario1\WpfClient » du repository « SimpleIdentityServer.Samples » et mettez à jour les propriétés dans les fichiers « UI\WpfClient\SecurityProxyClientApi » & « UI\WpfClient\Constants » & « Api\ClientApi\Startup ».

La page d’authentification est d’abord affichée. Dès lors que l’utilisateur avec le rôle « marketing » est authentifié alors les clients provenant de l’API sont listés.



Liste des clients :



## Second scénario : une API souhaite accéder à une ressource protégée

**Contexte** : Une équipe développe une nouvelle API qui récupère les données clientèles pour les enrichir avec leurs données marketing qui ont été récupérées depuis la création de l’entreprise comme par exemples : l’historique de navigation ou le nombre de plaintes.

**Problème** : Comment l’API peut-elle accéder à la liste des clients ?

**Solution** :

Le workflow ressemble beaucoup à celui du premier scénario. A la différence que cette fois-ci il n’est pas nécessaire de passer un identity-token au serveur UMA car il ne l’utilisera pas dans sa politique d’autorisation.

Récupérer RPT token

API

Service API

Serveur OPENID

Serveur UMA

Récupérer clients

Le processus d’implémentation est similaire à celui du premier scénario. Toutefois la tâche où la notion de resource owner intervient a été supprimée.

Identifier et catégoriser les entités

Ajouter un client

Ajouter une ressource

Développer

Ajouter politique d’autorisation

### Identifier et catégoriser les entités

Voici les entités que l’obtient après avoir utilisé le tableau de décision :

* Client : API
* Ressource : ClientApi / v1 / Clients / Get
* Liste des clients autorisés : API
* Liste des claims : aucun

### Ajouter un client

La plus part des paramètres peuvent facilement être déduits y-compris le grant type. Quel grant-type choisirez vous sachant qu’aucune interaction du resource owner n’est requise et qu’il y a une relation de confiance entre les deux APIs.

La bonne réponse est « client\_credentials », en effet vous n’encourez aucun risque de sécurité en le choisissant car le code est hébergé sur un serveur et devrait être inaccessible aux utilisateurs finaux.

Voici les paramètres que vous devez obtenir :

* Grant-types : client credentials
* Scopes : uma\_authorization et uma\_protection



### Ajouter une ressource

La ressource a déjà été créée dans le premier scénario.

### Ajouter la politique d’autorisation

Editez la politique d’autorisation de la ressource « resources\Apis\ClientApi\v1\ClientsController\Get » et ajoutez une nouvelle règle. Essayez de deviner les paramètres en vous appuyant sur la description du cas d’utilisation « Ajouter / Editer les règles d’autorisation ».

* Allowed clients : Marketing API
* Permissions : execute



### Développer

#### Modifier votre client API

Le Nuget package « SimpleIdentityServer.Proxy » doit être installé. Récupérez d’abord l’access token valide pour les scopes « uma\_protection » et « uma\_authorization » en utilisant le grant type « client credentials ».

private async Task<GrantedToken> GetAccessToken()

{

return await \_identityServerClientFactory.CreateTokenClient()

.UseClientSecretBasicAuth(\_clientId, \_clientSecret)

.UseClientCredentials("uma\_authorization", "uma\_protection")

.ResolveAsync("https://localhost:5443/.well-known/openid-configuration");

}

Passez ensuite le résultat en paramètre à la fonction ci-dessous afin de récupérer le RPT token. Ce dernier pourra ensuite être utilisé utilisé afin de récupérer la liste des clients.

public static async Task<string> GetRptToken(

string umaProtectionToken,

string umaAuthorizationToken)

{

var factory = new SecurityProxyFactory();

var proxy = factory.GetProxy(new SecurityOptions

{

UmaConfigurationUrl = "https://localhost:5445/.well-known/uma-configuration",

OpenidConfigurationUrl = "https://localhost:5443/.well-known/openid-configuration",

RootManageApiUrl = "https://localhost:5444/api"

});

try

{

var result = await proxy.GetRpt("resources/Apis/ClientApi/v1/ClientsController/Get", umaProtectionToken, umaAuthorizationToken, new List<string>

{

"execute"

});

return result;

}

catch (Exception ex)

{

return null;

}

}

Si vous voulez exécuter l’exemple complet ouvrez la solution « Scenario2\MarketingClient» du repository « SimpleIdentityServer.Samples » et mettez à jour les propriétés.

Lancez les deux projets « ClientApi » & « MarketingClient » puis naviguez vers l’url <http://localhost:5103/api/ratings>.



## Troisième scénario : Limiter l’accès à certaines fonctionnalités du site

**Contexte** : Les utilisateurs avec le rôle « administrator » peuvent voir le bouton « administrate » sur le site internet tandis que les autres ne le peuvent pas.

**Problème** : Comment limiter l’accès à certaines fonctionnalités selon le rôle des utilisateurs ?

**Solution** :

Comparé aux autres scénarios, le workflow présenté ci-dessous est assez simple :

* L’utilisateur s’authentifie auprès du serveur OPENID. Une fois authentifié alors les identity & access tokens sont retournés au site internet.
* Les permissions sont ensuite récupérées du serveur UMA et stockées dans un cookie. Les tokens qui ont été récupérés sont passés en paramètre de la requête.

Récupérer les permissions

Identity & access tokens

S’authentifie

Utilisateur

Serveur UMA

Site internet

Serveur OPENID

Comme les scénarios précédents nous allons suivre une méthodologie afin d’implémenter la solution.

Identifier et catégoriser les entités

Ajouter un client

Ajouter une ressource

Développer

Ajouter politique d’autorisation

Ajouter le rôle « administrator » au resource owner

### Identifier et catégoriser les entités

Voici les entités que l’obtient après avoir utilisé le tableau de décision :

* Client : WebSite
* Ressources : WebSite \ Scenario3 \ Home \ Admin
* Liste des clients autorisés : WebSite
* Liste des claims : rôle « administrator »

### Ajouter un client

Une fois de plus les paramètres sont assez faciles à déduire excepté le grant type. Le choix dépend encore une fois de la façon dont vous souhaitez authentifier les utilisateurs finaux. Si votre choix se porte sur le développement d’un formulaire « login » & « mot de passe » alors le grant type choisi sera « password ». Faîtes bien attention si vous choisissez ce grant-type car il est nécessaire d’avoir une relation de confiance avec le fournisseur d’identité et les identifiants du client « Id » & « secret » doivent être cachés du public.

Si au contraire vous souhaitez déléguer cette problématique d’authentification au serveur OPENID, choisissez alors le grant-type « implicit ». Lors de l’authentification l’utilisateur sera redirigé vers le serveur OPENID qui retournera ensuite les tokens. Notre préférence est toujours de choisir le grant-type le plus fiable et facile à implémenter, c’est pour cela que ce dernier a été sélectionné.

Faîtes bien attention de ne pas oublier les scopes obligatoires « uma\_authorization » & « uma\_protection » car il y a une interaction avec le serveur UMA.

Paramètres du client :

* Callback url : <https://localhost:5105/Authenticate/Callback>
* Grant types : implicit
* Response types : token & id\_token
* Scopes : uma\_protection, uma\_authorization, role, openid, profile



### Ajouter une ressource

Ajouter la ressource « WebSite \ Scenario3 \ Home \ Admin ».



### Ajouter la politique d’autorisation

Ajoutez une politique d’autorisation à la ressource. Essayez encore une fois de deviner les paramètres en vous appuyant sur le contexte ainsi que sur la description du cas d’utilisation : « Ajouter / Editer les règles d’autorisation ».

* Allowed clients : WebApplicationScenario3
* Permissions : read
* Allowed claims : rôle « administrator »



### Ajouter le rôle « administrator » au resource owner

. Editez les propriétés du resource owner de votre choix et ajoutez lui le rôle « administrator ».



### Développer

Je vous invite à jeter un œil au projet «Scenario3 \ WebApplication » du repository « SimpleIdentityServer.Samples ». Mettez à jour les paramètres du fichier « Constants.cs » et lancez l’application.

Si vous-vous connectez avec un utilisateur qui possède le rôle « administrator » alors la permission « Admin » est affichée, autrement il n’y a que trois permissions.



Permissions d’un utilisateur « administrator »



Permissions d’un utilisateur non « administrator »

# Documentation technique

# Reste à faire

|  |  |  |
| --- | --- | --- |
| Tâche | Priorité | Avancement |
| Ajouter un nouveau type de fournisseur d’identité SAML | 5 | DONE |
| Ajouter le code pour les fournisseurs d’identité OPENID | 5 | DONE |
| Afficher le bouton pour créer un fournisseur d’identité externe | 5 | DONE |
| Ajouter un fournisseur d’identité externe | 5 | DONE |
| Afficher le nom de l’utilisateur & son image & un bouton de déconnection | 5 | DONE |
| Vérifier le rôle de l’utilisateur | 5 | DONE |
| Vérifier pourquoi tous les fournisseurs d’identité ne sont pas affichés dans l’application WPF | 5 | DONE |
| Lorsque le client ne peut pas être authentifié alors retourner un message d’erreur lisible | 5 | DONE |
| Modifier le titre « GenerateResourceCommand » | 5 | DONE |
| Décrire le premier scénario | 5 | DONE |
| Afficher les clients | 5 | DONE |
| Utiliser un seul token pour plusieurs accès | 5 | DONE |
| Décrire le second scénario | 5 | DONE |
| Décrire le troisième scénario | 5 | DONE |
| Supprimer la propriété « include sub » de UmaIntrospection | 5 | DONE |
| Sécuriser l’API | 5 | TODO |

1. UMA terminologies : <https://docs.kantarainitiative.org/uma/rec-uma-core.html#terminology> [↑](#footnote-ref-1)
2. Autorisation : <https://docs.kantarainitiative.org/uma/rec-uma-core.html#claim-push> [↑](#footnote-ref-2)
3. Resource owner : <https://tools.ietf.org/html/rfc6749#section-1.1> [↑](#footnote-ref-3)
4. RFC OPENID : http://openid.net/specs/openid-connect-core-1\_0.html [↑](#footnote-ref-4)
5. RFC UMA : https://docs.kantarainitiative.org/uma/rec-uma-core.html [↑](#footnote-ref-5)
6. Règles de validation sur redirect\_uri : <https://openid.net/specs/openid-connect-registration-1_0.html#ClientMetadata> [↑](#footnote-ref-6)
7. Paramètres d’un client : <https://openid.net/specs/openid-connect-registration-1_0.html#ClientMetadata> [↑](#footnote-ref-7)
8. Paramètre request : <http://openid.net/specs/openid-connect-core-1_0.html#JWTRequests> [↑](#footnote-ref-8)
9. Documentation openid : <http://openid.net/specs/openid-connect-core-1_0.html> [↑](#footnote-ref-9)
10. RFC OpenId : https://openid.net/specs/openid-authentication-2\_0.html [↑](#footnote-ref-10)
11. RFC 6749 OAUTH2.0 : https://tools.ietf.org/html/rfc6749 [↑](#footnote-ref-11)
12. RFC WS-Federation : http://docs.oasis-open.org/wsfed/federation/v1.2/ws-federation.html [↑](#footnote-ref-12)
13. Implicit grant type : http://openid.net/specs/openid-connect-implicit-1\_0.html [↑](#footnote-ref-13)
14. Récupérer RPT token : https://docs.kantarainitiative.org/uma/rec-uma-core.html#rfc.section.3.5.1 [↑](#footnote-ref-14)
15. Introspection endpoint : https://docs.kantarainitiative.org/uma/rec-uma-core.html#rfc.section.3.4.1 [↑](#footnote-ref-15)
16. Liste des scopes : http://openid.net/specs/openid-connect-core-1\_0.html#ScopeClaims [↑](#footnote-ref-16)