Contents

[Benchmark 2](#_Toc455497613)

[Architecture 3](#_Toc455497614)

[Installation 5](#_Toc455497615)

[Etapes obligatoires 5](#_Toc455497616)

[Installer GIT 5](#_Toc455497617)

[Installer le certificat 5](#_Toc455497618)

[Installation manuelle 5](#_Toc455497619)

[Installer avec docker 5](#_Toc455497620)

[Prérequis 5](#_Toc455497621)

[Déploiement 6](#_Toc455497622)

[Site web 7](#_Toc455497623)

[Gérer les ressources 7](#_Toc455497624)

[Gérer les concepts « OpenId » et « Uma » 9](#_Toc455497625)

[Gérer les permissions 9](#_Toc455497626)

[Gérer la configuration 11](#_Toc455497627)

[Extensions Visual Studio 12](#_Toc455497628)

[Générer un proxy de sécurité 12](#_Toc455497629)

[Protéger les ressources 13](#_Toc455497630)

[Scénarios 17](#_Toc455497631)

[Protéger les actions de vos contrôleurs 17](#_Toc455497632)

[Protéger certaines parties de votre code 17](#_Toc455497633)

[Annexe 1 : convention de nommage 18](#_Toc455497634)

[Définitions 19](#_Toc455497635)

# Benchmark

Le tableau qui suit liste des différences entre les produits : Lokit, IdentityServer et le serveur Gluu.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Lokit** | **Identity Server** | **Gluu server** |
| Auteurs | Habart Thierry | Brock Allen & Dominick Baier | Société Gluu |
| Date début | Octobre 2015 | Janvier 2014 | Mars 2014 |
| Workflow Oauth2.0 |  | | |
| Client credentials | OK | OK | OK |
| Password | OK | OK | OK |
| Refresh token | OK | OK | OK |
| Workflow OpenId |  | | |
| Implicit | OK | OK | OK |
| Hybrid | OK | OK | OK |
| Autres fonctionnalités OpenId |  | | |
| Enregister un client ([RFC](https://openid.net/specs/openid-connect-registration-1_0.html)) | OK | NOK | OK |
| Signer ([JWS](https://tools.ietf.org/html/draft-ietf-jose-json-web-signature-41)) | OK | OK | OK |
| Encrypter ([JWE](https://tools.ietf.org/html/draft-ietf-jose-json-web-encryption-40)) | OK | NOK | NOK |
| Invalider la session | OK | OK | OK |
| Méthodes d’authentification d’un client ([RFC](http://openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication)) |  | | |
| client\_secret\_basic | OK | OK | OK |
| client\_secret\_post | OK | OK | OK |
| client\_secret\_jwt | OK | NOK | NOK |
| private\_key\_jwt | OK | NOK | NOK |
| none | OK | NOK | NOK |
| Différents modes de réponse |  | | |
| Query | OK | NOK | NOK |
| Fragment | OK | OK | OK |
| Form | OK | NOK | NOK |
| Quelques paramètres OpenId |  | | |
| claims ([RFC](http://openid.net/specs/openid-connect-core-1_0.html#ClaimsParameter)) | OK | NOK | NOK |
| request ([RFC](http://openid.net/specs/openid-connect-core-1_0.html#RequestObject)) | OK | NOK | NOK |
| Qualité |  | | |
| Code coverage | 84% | ?? | ?? |
| Nombre UTs | 800 | 400 | ?? |
| UMA ([RFC](http://openid.net/specs/openid-heart-uma-2015-12-09.html)) |  | | |
| UMA implémenté | OK | NOK | OK |
| UI |  | | |
| UI présente | OK | OK | OK |
| CRUD openid assets (client, scope, resource owner etc ..) | OK | OK | OK |
| CRUD uma assets (resource, authorization policy etc …) | OK | NOK | OK |
| Ressource organisée par URL | OK | NOK | NOK |
| Activer ou désactiver des identity providers | OK | NOK | OK |
| Déploiement |  | | |
| Méthodes de déploiement | Docker ou manuellement | Manuellement (nécessite la création d’un projet web) | Manuellement en suivant le guide |
| Autres |  | | |
| Nombre certifications | 5 | 4 | 5 |
| Langage préféré | C# | C# | Java |
| Outils pour interagir aisément avec les APIs | Extensions Visual Studio et nuget packages | Nuget package | NOK |

Même si Lokit offre un bon nombre de fonctionnalités face à ses concurrents, il tire sa force de part sa facilité de déploiement et d’utilisation. En effet un développeur .NET qui n’est pas familier avec les notions OpenId et UMA, peut aisément utiliser le produit grâce aux différents outils mis à disposition.

La partie qui suit décrit brièvement l’architecture et les différents composants.

# Architecture

Le schéma ci-dessous montre les interactions entre les composants.

Configu ration API

Manager API

Uma API

Openid API

WebSite API

WebSite

UMA DB

OpenId DB

Configuration DB

Extension Visual Studio

WebSite API DB

Sur une totalité de 11, il y a 5 APIs, une extension Visual Studio, un site web et quatre base de données :

* Manager API : il est utilisé afin d’exécuter des operations « CRUD » sur les concepts d’OpenId tels que les clients ou les utilisateurs.
* OpenId & UMA Api : comme leurs noms l’indiquent, ce sont l’implémentation des RFC OpenId et UMA.
* Configuration API : gère la configuration de l’API OpenId, par exemple : activer ou désactiver certaines méthodes d’authentification.
* WebSite API :couche d’abstraction qui ajoute le concept de URI aux ressources.
* WebSite : il est utilisé par un administrateur afin de gérer les accès à certaines ressources.
* Extension Visual Studio : il est utilisé par un développeur .NET afin d’interagir avec les différents composants.

Deux rôles ont été identifiés « administrateur » et « développeur », leurs uses cases sont décrits dans le schéma ci-dessous :

<<include>>

<<include>>

<<include>>

<<include>>

<<include>>

Développeur

Administrateur

<<include>>

Les « uses cases » sont décrits plus tard après le chapitre d’installation.

# Installation

Le guide part du principe que tous les composants sont installés sur la même machine, mais gardez à l’esprit qu’il est tout à fait possible de les déployer séparément.

## Etapes obligatoires

Que vous choisissez d’installer le produit manuellement ou par Docker, il y a quelques prérequis à installer sur votre machine.

### Installer GIT

GIT doit être installé afin de récupérer les sources du projet, suivez le guide du site officiel : ([link](https://git-scm.com/downloads))



### Installer le certificat

Le certificat « LokitCA.cer » joint au produit doit être installé sur votre machine dans « Certificate Store \ Local User \ Trusted CA ».



## Installation manuelle

TODO

## Installer avec docker

### Prérequis

Le produit peut être déployé dans plusieurs conteneurs Docker. La procédure est assez simple à suivre et se résume en quelques étapes. Mais avant de commencer vous devez vous assurer que les prérequis suivants ont bien été installés et configurés sur votre machine.

#### Installer Docker

Docker doit être installé, vous pouvez suivre le guide sur le site officiel ([lien](https://docs.docker.com/engine/installation/)) qui est très bien expliqué :



#### Configurer VirtualBox

Par défaut Docker utilise VirtualBox pour lancer sa machine virtuelle. Des règles de redirection de ports doivent être ajoutées sur la machine « default », sans quoi vous ne pourrez pas naviguer sur le site :

* Ouvrez VirtualBox et sélectionnez la machine « default »
* Cliquez sur « Configuration » puis sélectionnez l’onglet « network »
* Cliquez sur le bouton « port redirection » et assurez-vous de bien avoir les mêmes règles :



### Déploiement

Maintenant que vous avez fini avec les prérequis, vous pouvez lancer l’installation.

#### Récupérer les sources

Avec GIT récupérez les sources du dépôt SimpleIdentityServerDocker ([link](https://github.com/thabart/SimpleIdentityServerDocker.git)) puis ouvrez un invité de commande et naviguez vers le nouveau répertoire.

#### Lancer le déploiement

Toujours dans le même invité de commande, exécuter l’instruction « *docker-compose* up » et attendez que le produit se lance.

Une fois l’application installée et déployée, vous pouvez en tant qu’administrateur vous connecter au site web.

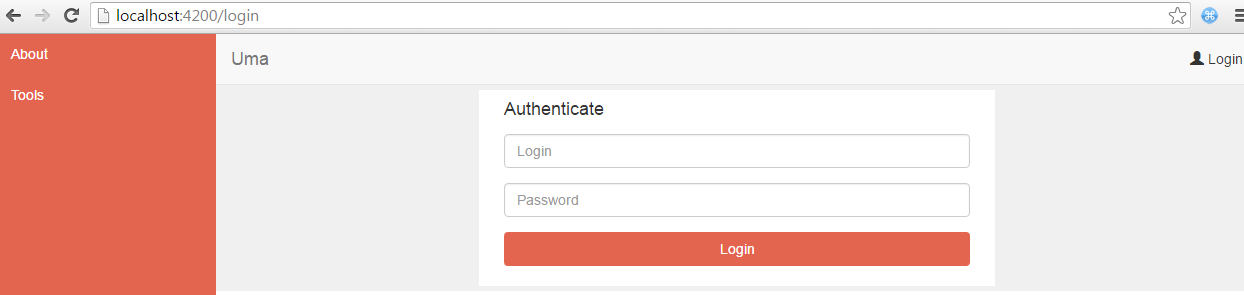
# Site web

Ouvrez dans votre navigateur préféré le site <http://localhost:4200> et connectez vous en tant qu’administrateur. Le login et mot de passe par défaut sont :

* Login : administrator
* Password : password

Il est vivement conseillé de modifier ces informations afin d’éviter toute intrusion.

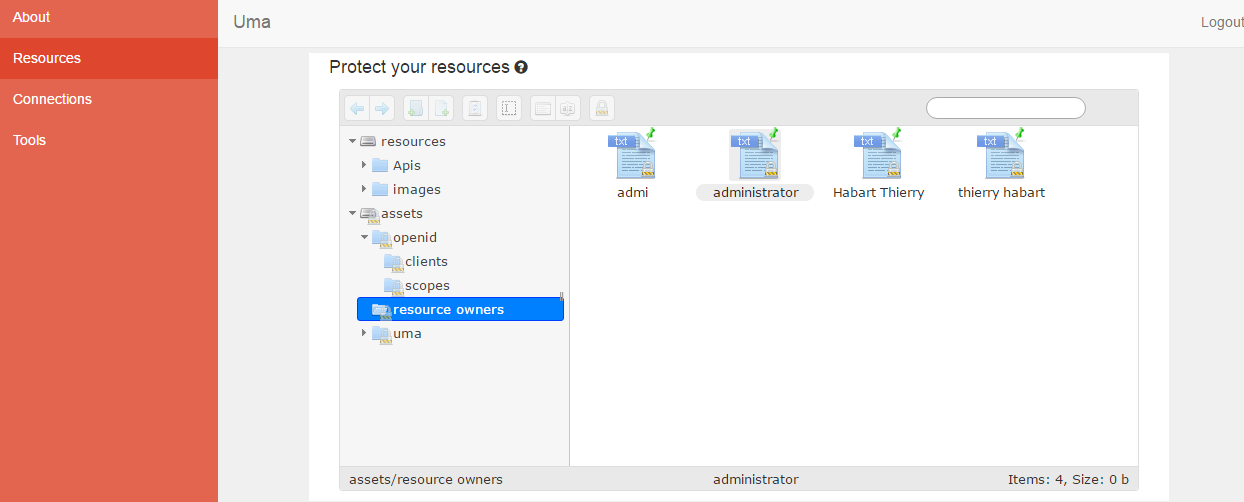
**Note** : Un administrateur est n’importe quel utilisateur qui possède le rôle « administrator »



## Gérer les ressources (UC1)

Lorsque vous êtes connecté, allez dans l’onglet « resources ». Un explorateur de fichiers s’affiche, c’est à cet endroit que vous pouvez gérer les concepts clés d’« UMA » & « OpenId » ainsi que les ressources.

Définition : Une ressource est quelque chose que vous souhaitez protégé, ça peut être l’opération d’un contrôleur ou l’image d’un utilisateur



La gestion des ressources est accessible en cliquant sur le nœud « resources » dans l’arborescence. Il en existe deux types  : fichier ou dossier. La différence entre les deux est que le dossier peut contenir une ou plusieurs ressources tandis qu’un fichier ne peut pas en contenir.

Resource

Dossier

Fichier

Est-une

Est-une

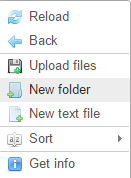
Contient

Est-fils de

0…\*

0…1

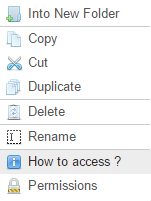
Pour ajouter une ressource, affichez le « context menu » du répertoire courant en cliquant droit dessus, puis choisissez entre créer un dossier ou fichier.



Les ressources ont été organisées de cette façon afin de simplifier l’accessibilité, mais aussi de rendre facile la gestion des permissions (en effet les ressources héritent des permissions de leurs parents).

Des actions peuvent être exécutées individuellement sur chaque ressource pour cela sélectionnez en une et affichez son « context-menu » . Vous retrouverez certaines des fonctionnalités d’un explorateur de fichiers comme : copier, couper, dupliquer, renommer mais aussi de nouvelles :

* How to access : explique comment accéder à cette ressource
* Permissions : affiche une fenêtre pour gérer les permissions



Vous remarquerez que si vous affichez le context-menu d’une ressource avec un « pin » en haut à gauche, alors toutes les actions ne sont pas visibles. Ce comportement est normal, car un utilisateur ne peut pas exécuter les actions normales sur une ressource qui est protégée. Il en va de même pour toutes les ressources dans « assets ».

Le composant graphique offre beaucoup plus de fonctionnalités dont la plus part sont inspirées de l’explorateur de fichiers Windows. En voici quelques unes qui peuvent vous être bien utiles, si vous souhaitez obtenir la liste des raccourcis cliquez sur « F1 »

|  |  |
| --- | --- |
| Description | ShortCut |
| Rechercher une ressource dans le répertoire courant | Ctrl+s |
| Afficher la liste des permissions | Ctrl+P |
| Copier et coller plusieurs ressources | Sélectionner plusieurs ressources et Cltr+C et Cltr+V |
| Sélectionnez une ou plusieurs ressources puis supprimer les | Sélectionner plusieurs ressources et cliquez sur « Delete » |

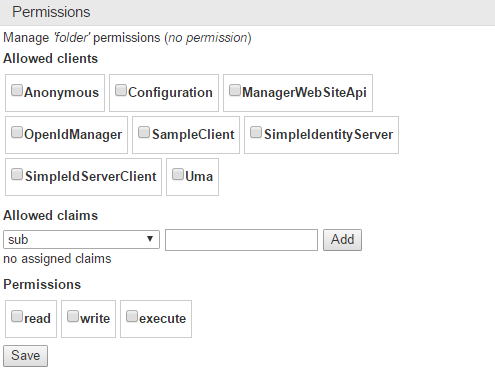
## Gérer les permissions (UC2)

La gestion des permissions est le « use-case » le plus important du produit, sans lui les ressources ne peuvent pas être protégées. Une permission est définie par trois paramètres :

* Clients : applications autorisées
* Claims : utilisateurs autorisés
* Scopes : actions qui peuvent être exécutées.

Vous définissez pour une ressources et ses fils, une politique d’autorisation commune dans laquelle certaines applications et utilisateurs peuvent exécuter des opérations.

Pour définir une permission, sélectionnez une ressource, affichez le « context-menu » et cliquez sur « Permissions ». Une nouvelle fenêtre s’affiche avec les paramètres requis pour définir la politique d’autorisation.

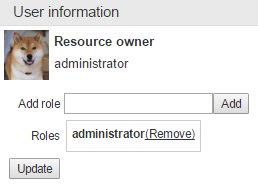


Sélectionnez les clients et ajoutez des claims pour lesquels vous souhaitez autoriser l’accès pour les actions « read », « write » ou « execute ».

L’accès peut être restreint selon les rôles de l’utilisateur, mais étant donné que ce claim n’est pas conforme au standard OpenId, il doit alors être défini.

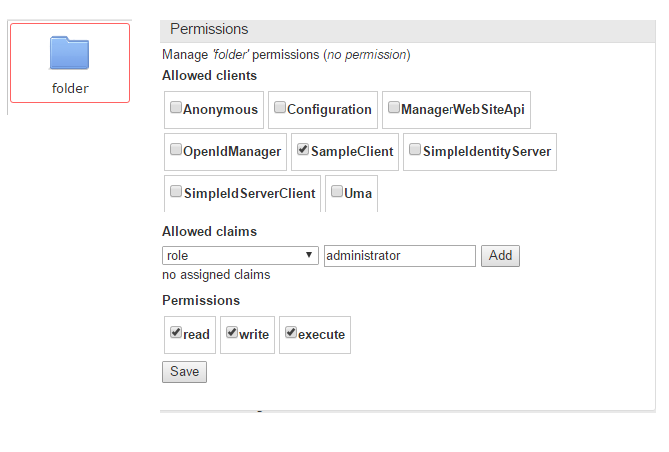
**Attention** : Cette procédure ne fonctionne que pour des utilisateurs locaux. Pour plus d’information sur la différence entre un compte externe et locale voir le chapitre XXXX

Pour ajouter un rôle, naviguez vers « assets \ resource owners », sélectionnez l’utilisateur pour lequel vous souhaitez ajouter le rôle, affichez son « context-menu » et cliquez sur « Resource owner info »



Ajouter le rôle et confirmer la mise à jour en cliquant sur « Update ».

Maintenant que le rôle a été défini pour l’utilisateur « administrator », vous pouvez réutiliser sa valeur et confirmer la politique d’autorisation en cliquant sur « update ».



A la fin de l’opération, la ressource qui a la permission devrait être encadrée en rouge.

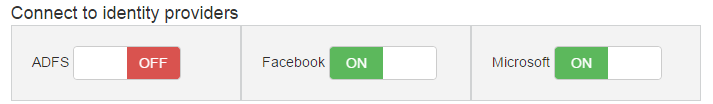
## Gérer les concepts « OpenId » et « Uma » (UC3)

La gestion des concepts est très similaire à ce qui a été expliqué dans précédente partie. Commencez par sélectionner le nœud « assets » et choisissez entre « openid » et « uma ». A partir de là vous pourrez effectuer des opérations de « mise à jour », « création » et « suppression » .

Faites bien attention à ce que vous faîtes, ne supprimez pas n’importe quel scope ou client dans quel cas certaines de vos applications ne fonctionneront plus correctement.

## Gérer la configuration (UC4)

Pour l’instant la configuration est utilisée uniquement pour activer / désactiver les fournisseurs d’identités externes et mettre à jour leurs paramètres.



Les paramètres peuvent être mis à jour en sélectionnant l’un des Identity Providers.

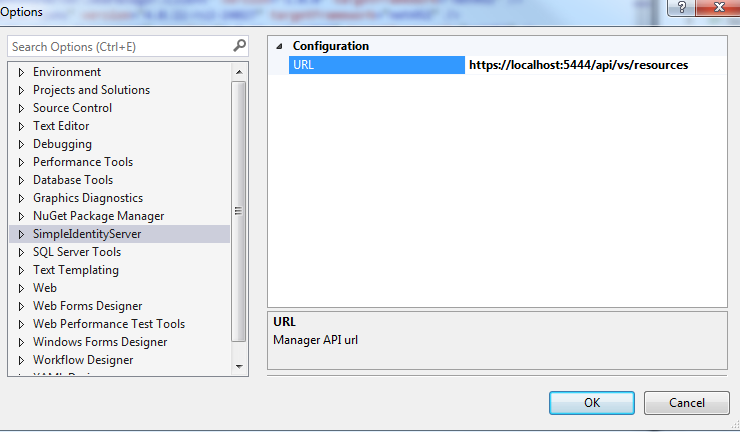
**Attention** : Seule la version 2016 d’ADFS est supportée. Les versions antérieures seront supportées dans les prochaines releases.

# Extension Visual Studio

Attention l’extension Visual Studio fonctionne pour le moment que sur la version 2015. D’autres versions seront supportées à l’avenir.

Téléchargez l’extension et installez là en double cliquant dessus. Lancez Visual Studio et dans la fenêtre des options « Tools > Options > SimpleIdentityServer », mettez à jour l’URL si nécessaire.

Laissez la valeur par défaut si vous avez tout déployé sur votre machine et que vous n’avez pas touché à la configuration.

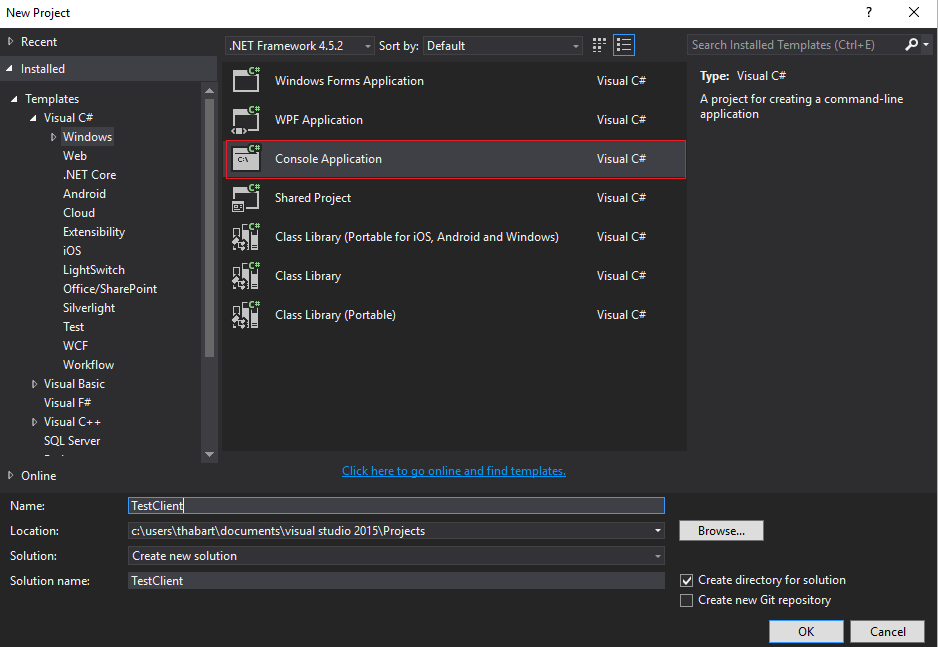


Une fois installée vous verrez deux nouvelles actions dans le « context menu » du projet : « Generate security proxy » et « Generate resource ». Leurs fonctions sont expliquées dans les parties suivantes.

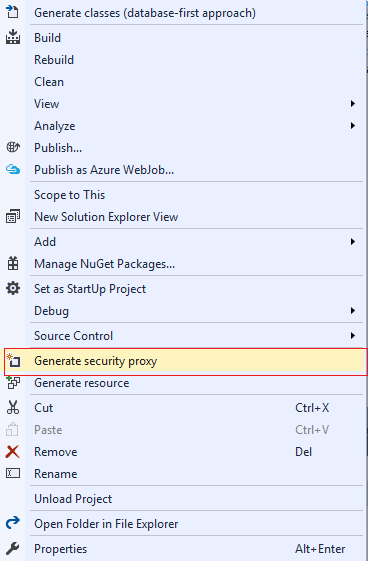
## Générer un proxy de sécurité (UC5)

**Objectif** : Récupérer un token qui sera utilisé afin d’interagir avec une ressource protégée.

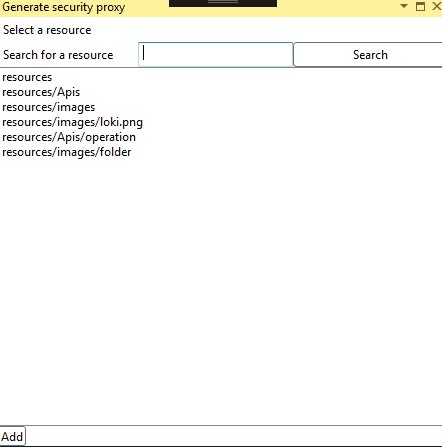
Créer une nouvelle application console. Dans la fenêtre « new project » sélectionnez le template «  Visual C# \ Console Application » puis saisissez le nom du projet « TestClient », confirmez la création en appuyant sur OK.



Lorsque le projet est créé et compilé, ouvrez le « context menu » du projet « TestClient » et exécutez l’action « Generate security proxy »



Une nouvelle fenêtre s’affiche. Attendez quelques instant le temps que la liste des ressources s’affiche.



Sélectionnez la ressource que vous souhaitez accéder, puis cliquer sur « Add ».

Les opérations suivantes seront effectuées :

**Note** : Les nuget package suivant sont installés : SimpleIdentityServer.UmaIntrospection.Authentication & SimpleIdentityServer.Uma.Authorization

Vous pouvez ensuite utiliser le nouveau « SecurityProxy » afin de récupérer un « RPT » token en appelant soit « GetRptToken » ou « GetRptToken(string idToken) ».

Si vous ne disposez pas d’un « identity token », vous pouvez toujours le récupérer en appelant la méthode « AuthProvider.GetIdentityToken ».

N’oubliez pas de mettre à jour les propriétés des options : « AuthOptions » et « SecurityOptions » ainsi que les paramètres.

|  |  |
| --- | --- |
| Propriété | Description |
| AuthOptions \ OpenIdConfigurationUrl | Discovery endpoint de OpenId |
| AuthOptions \ ClientId | L’identifiant et le secret du client qui a accès au scopes : « openid », « role » et « profile » et qui autorise le grant type : « password » |
| AuthOptions \ ClientSecret |
| SecurityOptions\ClientId | Identifiant et secret du client qui a accès au scopes « uma\_authorization » & « uma\_protection »et qui autorise le grant type : « client\_credentials » |
| SecurityOptions\ClientSecret |
| SecurityOptions\UmaConfigurationUrl | Discovery endpoint de UMA |
| SecurityOptions\OpenIdConfigurationUrl | Discovery endpoint de OpenId |
| SecurityOptions\RootManageApiUrl | L’URL de « website API » |

Récupérez ensuite le « RPT » token et passez le dans le header de votre requête HTTP :

var token = SecurityProxy\_5774fa6a35e6e9043b68eabd.GetRptToken("<id \_token>");

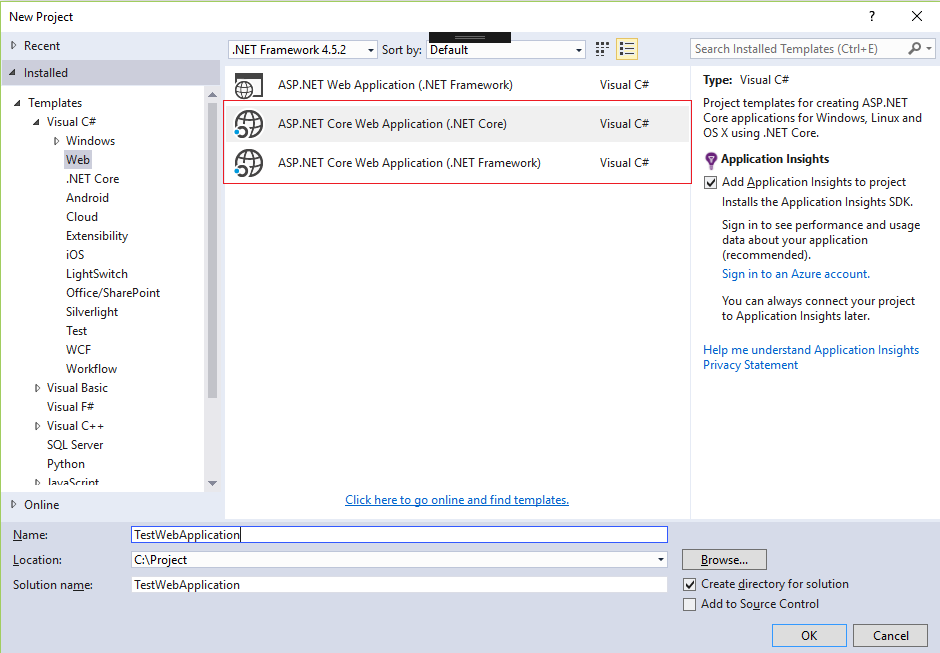
var request = new HttpRequestMessage();

request.Headers.Add(“Authorization”, “Bearer : “ + token);

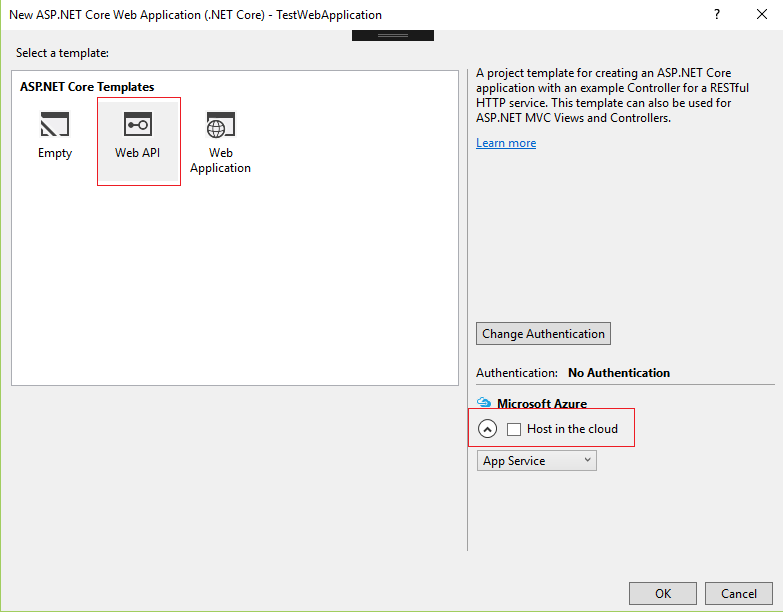
## Protéger les ressources (UC6)

**Objectif** : sélectionner et protéger les actions des contrôleurs d’un projet WEB.API.

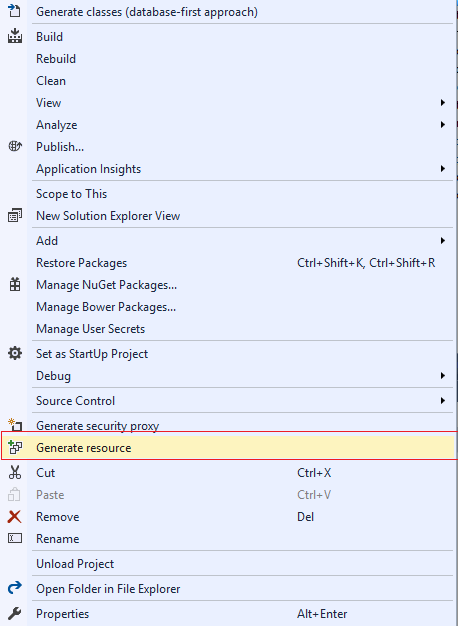
Créer une nouvelle solution avec un projet ASP.NET CORE WEB API. Pour cela dans la fenêtre « new project » sélectionnez le template « *ASP.NET Core Web Application (.NET CORE*) » ou « *ASP.NET Core Web Applications (.NET Framework)* ».  
Rentrez le nom du projet « TestWebApplication » puis cliquez sur « OK »



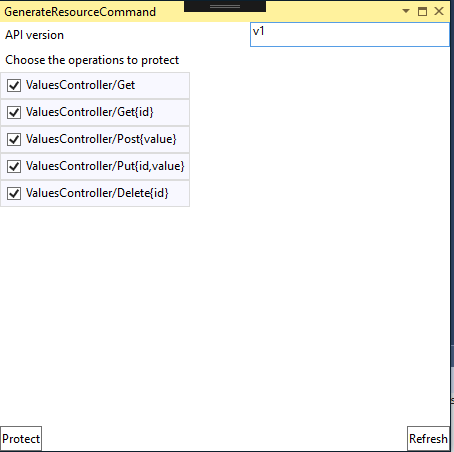
Dans la fenêtre suivante sélectionnez le type de projet « WEB.API », décochez « host in the cloud » et confirmer la création en cliquant sur « OK ».



Lorsque votre projet est créé et compilé, vous pouvez commencer à protéger les méthodes des contrôleurs. Pour cela sélectionnez votre projet « TestWebApplication », affichez le « context menu » et sélectionnez l’option « generate resource »



Une nouvelle fenêtre Visual Studio s’affiche. Attendez quelques secondes le temps que les opérations des contrôleurs soient listées. Si aucune opération n’est visible alors cliquez sur le bouton « refresh » et ré attendez quelques secondes.



Sélectionnez les opérations que vous souhaitez protéger, spécifier le numéro de version de votre API et cliquez sur « Protect ».

L’opération effectuera plusieurs tâches :

**Note** : S’il y a eu une erreur durant l’installation des packages, vérifiez que le feed a bien été ajouté. Deux Nuget packages doivent être installés :

🡺 SimpleIdentityServer.UmaIntrospection.Authentication (version 1.0.0)

🡺 SimpleIdentityServer.Uma.Authorization (version 1.0.0)

Si vous n’avez observé aucune erreur alors vous pouvez modifier votre fichier « Startup.cs » comme il est expliqué dans « Startup\_Sample.cs ».

Activer l’authentification en ajoutant dans la méthode « Startup.cs\Configure » le code suivant :

var options = new UmaIntrospectionOptions

{

ResourcesUrl = "http://localhost:8080/api/vs/resources",

UmaConfigurationUrl = "http://localhost:5001/.well-known/uma-configuration"

};

app.UseAuthenticationWithUmaIntrospection(options);

N’oubliez pas de vérifier et mettre à jour les valeurs des propriétés « ResourcesUrl » & « UmaConfigurationUrl » avec les bonnes URLs.

Vous pouvez choisir entre activer l’autorisation par convention ou par ressource.

La première se base sur une convention de nommage (voir Annexe 1) tandis que pour l’autre l’URL de la ressource doit être passée en paramètre.

services.AddAuthorization(options =>

{

options.AddPolicy("uma", policy => policy.AddConventionalUma());

});

services.AddAuthorization(options =>

{

options.AddPolicy("resourceSet", policy => policy.AddResourceUma("values/get", new List<string>

{

"execute"

}));

});

Afin de rendre effectif l’autorisation, les opérations ou contrôleurs doivent être décorées par l’attribut [Authorization] et le nom de la politique (resourceSet ou uma) doit être passé en paramètre.

[Authorize("resourceSet")]

public IEnumerable<string> Get()

{

return new string[] { "value1", "value2" };

}

# Scénarios

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

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

## 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 group « Active Directory : 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 étapes :

* Récupérer un Identity Token avec le grant type : « implicit ». Le token est passé comme paramètre au callback une fois que l’utilisateur est authentifié.
* L’application WPF récupère un RPT Token en passant en paramètre ses « credentials » et l’Identity Token de l’utilisateur.
* Lorsque le RPT Token est reçu alors les clients sont retournés par le service API.

Application WPF

Service API

Serveur autorisation

Récupérer Identity Token

Identity token

Serveur autorisation

Authentifier utilisateur

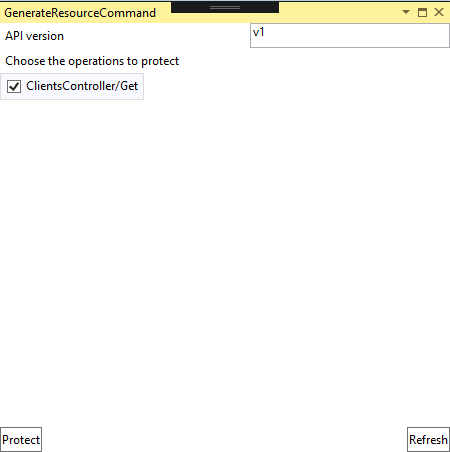
Récupérer RPT token

RPT token

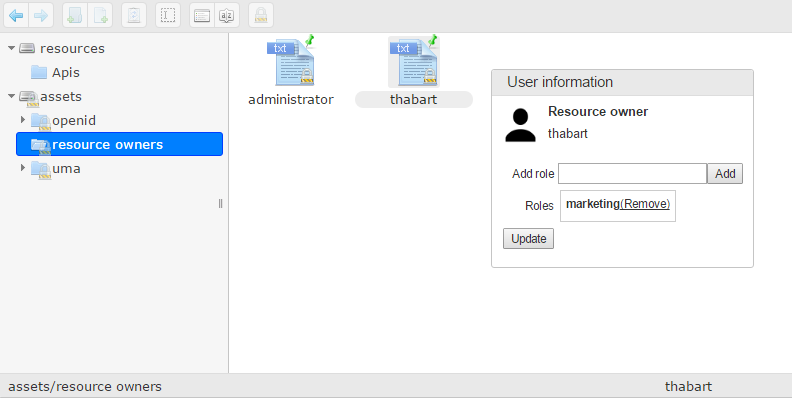
Récupérer clients

Etapes suivies pour implémenter la solution :

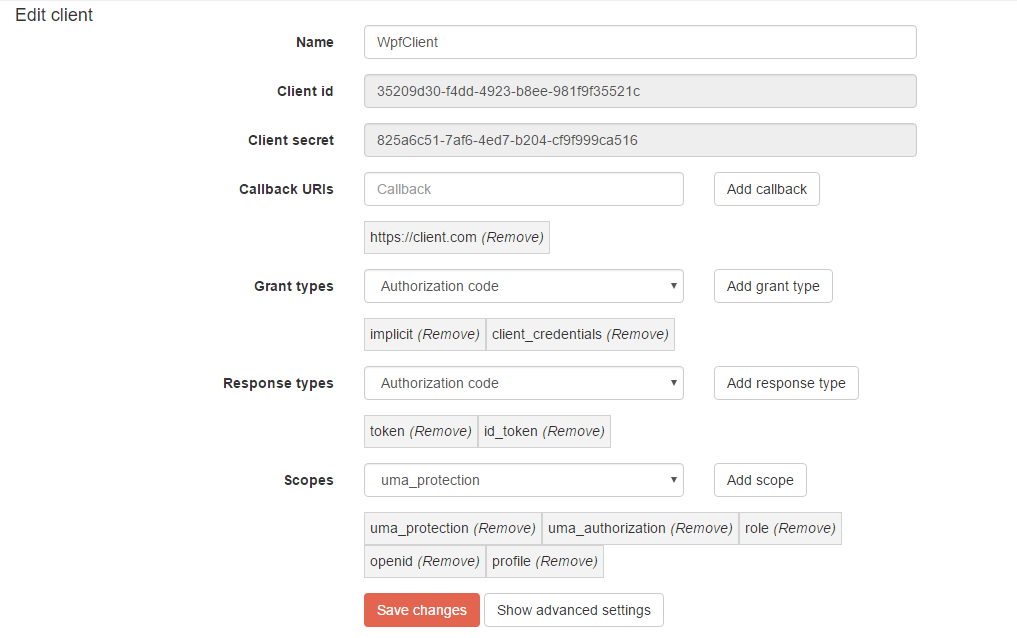
1. Créer deux applications : lourde (WPF) et un service web API
2. Ajoutez un contrôleur « ClientController » et une méthode «Get » qui retourne la liste des clients.
3. Sélectionnez le projet et cliquez sur l’option « Generate resources ». Une fois l’opération affichée protégez la en cliquant sur « protect ».



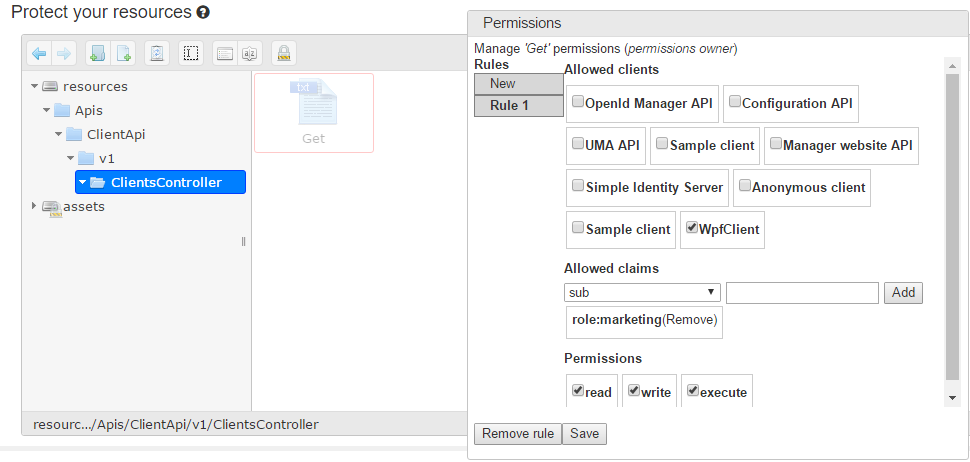
1. Connectez-vous sur le site <http://localhost:4200> en tant qu’administrateur
2. Sélectionnez un utilisateur et assignez-lui le rôle « marketing ». Vous utiliserez cet utilisateur lorsque vous vous connecterez à l’application.



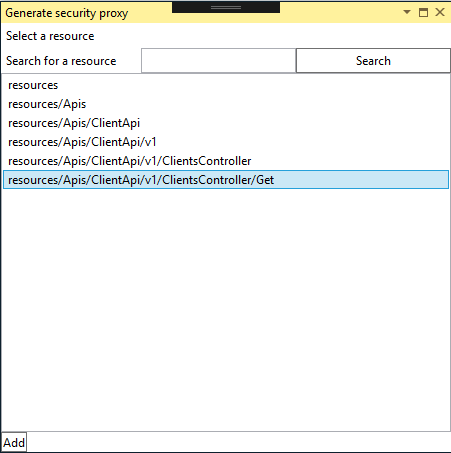
1. Créer un nouveau client et éditez ses propriétés :
   1. Grant type : implicit, client credentials
   2. Scopes : openid, role, profile, uma\_authorization, uma\_protection



1. Sélectionnez la ressource « resources/Apis/ClientApi/v1/ClientsController/Get » et éditez ses permissions. Ajoutez une nouvelle règle d’autorisation valide pour le nouveau client (ajouté à l’étape 6) et le claim « role : marketing »



1. Ouvrez votre projet WPF, sélectionnez-le et cliquez sur « Generate security proxy ». Attendez quelques secondes et sélectionnez la ressource qui a été ajoutée : « ClientController\Get » puis cliquez sur « Add »



1. Utilisez la méthode de la classe « AuthProvider » pour récupérer l’Identity Token et passez-le en paramètre à la méthode « SecurityProxy\GetRptToken »
2. Le Token peut être passé dans «Autorisation header » de votre requête afin de récupérer les clients.

## API souhaite accéder à une opération protégée

*Ce scénario ressemble beaucoup au précédent et ne possède pas beaucoup de différences*

TODO

## Rendre visible certaines parties de l’application web

Pour implémenter ce scénario il faut :

* Développer une nouvelle opération qui retourne toutes les ressources accessibles
* Supporter le caching au niveau UMA

TODO

# Annexe 1 : convention de nommage

**Le patron** : resources /Apis / <nom du projet> / <numéro de version> / <nom du contrôleur > / <nom de l’action> ({liste des paramètres})

# Définitions

Une ressource est quelque chose que vous souhaitez protégé, ça peut être l’opération d’un contrôleur ou l’image d’un utilisateur