Exemple C#

# Prérequis

Pour tester l’exemple C#, il est nécessaire d’installer un environnement de développement intégré IDE : Microsoft Visual Studio. Lien de téléchargement : https://visualstudio.microsoft.com/fr/

# Création d’un projet Visual Studio

Dans Visual Studio, créez un nouveau projet et choisissez « Application console (.NET Core). Cliquez sur « suivant », puis entrez un nom et choisissez un emplacement pour votre projet, puis cliquez sur « créer ».

Ensuite, dans « Explorateur de solutions », clic droit sur votre projet 🡺Modifier le fichier projet, et remplacez la ligne :

<TargetFramework>netcoreapp3.1</TargetFramework>

Par :

<TargetFramework>net48</TargetFramework>

# Inclure les DLL

Dans l’explorateur de solutions à droite, faites clic droit sur votre projet 🡺 Ajouter 🡺 Nouveau dossier 🡺 appeler le dossier « libs ».

Copiez-collez les DLL présentes dans le dossier KalimaCSharpExample/libs dans le vôtre. La plupart de ses DLL proviennent de ikvm :

<https://github.com/jessielesbian/ikvm>

IKVM est une implémentation de java pour Microsoft .NET, si vous avez besoin de plus de fonctionnalités java, veuillez installer ikvm et inclure les DLL nécessaires dans votre projet.

Ensuite, dans visual studio, faites clic droit sur votre projet 🡺 Ajouter 🡺 Reference de projet 🡺 Parcourir 🡺 Sélectionnez toutes les DLL que vous avez ajouter dans votre dossier libs.

# KalimaClientCallBack

Créez une nouvelle classe sous le nom « KalimaClientCallBack » :

* Clic droit sur votre projet
* Ajouter
* Nouvel élément
* Sélectionnez « classe » et entrez « KalimaClientCallBack.cs » dans le champ « nom »
* Ajouter

Ci-après le code complet de la classe (le namespace peut changer suivant le nom de votre projet) :

A ce stade, le projet comporte des erreurs qui seront corrigées plus tard.

**using** System**;**

**using** org**.**kalima**.**kalimamq**.**message**;**

**using** org**.**kalima**.**kalimamq**.**nodelib**;**

**using** org**.**kalima**.**kalimamq**.**netlib**;**

**using** org**.**kalima**.**cache**.**lib**;**

**using** java**.**util**.**concurrent**.**atomic**;**

**using** java**.**nio**.**channels**;**

**namespace** KalimaCSharpExample

**{**

**public** class KalimaClientCallBack **:** ClientCallback

**{**

**private** Client client**;**

**private** Logger logger**;**

**private** Node node**;**

**public** KalimaClientCallBack **(**Client client**)**

**{**

**this.**client **=** client**;**

**this.**node **=** client**.**getNode **();**

**this.**logger **=** node**.**getLogger **();**

**}**

**public** void putData**(**SocketChannel ch**,** KMessage msg**)** **{**

KMsg kMsg **=** KMsg**.**setMessage **(**msg**);**

client**.**getClone **().set** **(**kMsg**.**getCachePath**(),** kMsg**,** **true,** **false);**

**}**

**public** void onConnectionChanged**(**int status**,** NioClient nioClient**){**

logger**.**log\_srvMsg **(**"ExampleClientNode"**,** "KalimaClientCallBack"**,** Logger**.**DEBUG**,** "onConnectionChanged status=" **+** status**);**

client**.**getClone**().**onConnectedChange**(**status**==**Node**.**CLIENT\_STATUS\_CONNECTED**)** **?** **new** AtomicBoolean**(true)** **:** **new** AtomicBoolean**(false),** nioClient**);**

**}**

**public** void onCacheDeleted **(**string cacheSubPath**){**

logger**.**log\_srvMsg **(**"ExampleClientNode"**,** "KalimaClientCallBack"**,** Logger**.**DEBUG**,** "onCacheDeleted cacheSubPath=" **+** cacheSubPath**);**

**}**

**public** void putRequestData**(**SocketChannel ch**,** KMessage msg**)** **{}**

**public** void onNewVersion**(**int majver**,** int minver**)** **{}**

**public** void onNewCache**(**String cachePath**)** **{}**

**}**

**}**

# KalimaServerCallBack

De la même manière, ajoutez une classe KalimaServerCallBack, ci-après le code complet :

**using** System**;**

**using** org**.**kalima**.**kalimamq**.**nodelib**;**

**using** org**.**kalima**.**kalimamq**.**message**;**

**using** java**.**nio**.**channels**;**

**namespace** KalimaCSharpExample

**{**

**public** class KalimaServerCallBack **:** ServerCallback

**{**

**public** KalimaServerCallBack**()** **{** **}**

**public** void putData**(**SocketChannel socket**,** KMessage kMessage**)** **{** **}**

**public** void processSnapshot**(**SocketChannel ch**,** KMessage kMessage**)** **{** **}**

**public** void disconnected**(**String name**,** int port**)** **{** **}**

**public** void nodeAcked**(**String nodeName**,** String hostName**,** int port**)** **{** **}**

**public** void nodeDisconnected**(**Node node**,** bool b**)** **{** **}**

**public** void nodeConnected**(**Node node**)** **{** **}**

**public** void onConnectionChanged**(**int status**)** **{** **}**

**public** void addCache**(**String cachePath**)** **{** **}**

**public** void removeCache**(**String cachePath**)** **{** **}**

**public** void processSnapshot**(**SocketChannel soc**,** String cachePathSeqs**)** **{** **}**

**public** KMessage processNextMessage**(**KMessage msg**,** SocketChannel soc**)**

**{**

**return** **null;**

**}**

**public** void onVoteDone**(**bool isLeader**)** **{** **}**

**}**

**}**

# Client

Enfin, renommez votre classe principale (par défaut Program.cs) en Client.cs 🡺 Clic droit sur Program.cs 🡺 Renommer.

Ci-après le code complet de la classe :

**using** System**;**

**using** org**.**kalima**.**kalimamq**.**nodelib**;**

**using** org**.**kalima**.**kalimamq**.**message**;**

**using** org**.**kalima**.**kalimamq**.**crypto**;**

**using** org**.**kalima**.**cache**.**lib**;**

**using** ikvm**.**extensions**;**

**namespace** KalimaCSharpExample

**{**

**public** class Client **:** KalimaNode

**{**

**private** Node node**;**

**private** Clone clone**;**

**private** Logger logger**;**

**private** KalimaServerCallBack kalimaServerCallback**;**

**private** KalimaClientCallBack kalimaClientCallback**;**

**private** ClonePreferences clonePreferences**;**

**private** byte**[]** devId**;**

**public** static void Main**(**string**[]** args**)**

**{**

**try**

**{**

Client client **=** **new** Client**(**args**);**

**}**

**catch** **(**Exception e**)**

**{**

e**.**printStackTrace**();**

**}**

**}**

**public** Client**(**string**[]** args**)**

**{**

clonePreferences **=** **new** ClonePreferences**(**args**[**0**]);**

logger **=** clonePreferences**.**getLoadConfig**().**getLogger**();**

initComponents**();**

System**.**Threading**.**Thread**.**Sleep**(**2000**);**

**for** **(**int i **=** 0**;** i **<** 10**;** i**++)**

**{**

String body **=** "hello" **+** i**;**

KMsg kMsg **=** **new** KMsg**(**0**);**

node**.**sendToNotaryNodes**(**kMsg**.**getMessage**(**devId**,** KMessage**.**PUB**,** "/sensors"**,** "key" **+** i**,** body**.**getBytes**(),** **new** KProps**(**"10"**)));**

System**.**Threading**.**Thread**.**Sleep**(**1000**);**

**}**

**}**

# Fichier de configuration

**public** void initComponents**()**

**{**

byte**[]** key **=** **new** byte**[]** **{**

**(**byte**)**0x20**,** **(**byte**)**0xf7**,** **(**byte**)**0xdf**,** **(**byte**)**0xe7**,**

**(**byte**)**0x18**,** **(**byte**)**0x26**,** **(**byte**)**0x0b**,** **(**byte**)**0x85**,**

**(**byte**)**0xff**,** **(**byte**)**0xc0**,** **(**byte**)**0x9d**,** **(**byte**)**0x54**,**

**(**byte**)**0x28**,** **(**byte**)**0xff**,** **(**byte**)**0x10**,** **(**byte**)**0xe9

**};**

devId **=** KKeyStore**.**setDevId**(**clonePreferences**.**getLoadConfig**().**getFilesPath**(),** key**,** logger**);**

node **=** **new** Node**(**clonePreferences**.**getLoadConfig**());**

node**.**setDevID**(**devId**);**

clone **=** **new** Clone**(**clonePreferences**,** node**);**

kalimaServerCallback **=** **new** KalimaServerCallBack**();**

kalimaClientCallback **=** **new** KalimaClientCallBack**(this);**

node**.**connect**(**kalimaServerCallback**,** kalimaClientCallback**);**

**}**

**public** Node getNode**()**

**{**

**return** node**;**

**}**

**public** Logger getLogger**()**

**{**

**return** logger**;**

**}**

**public** Clone getClone**()**

**{**

**return** clone**;**

**}**

**}**

Ajoutez un dossier cfg dans votre projet, et placer dedans un fichier « node.config ». Ce fichier de configuration sera passé en paramètre de l’application.

Voici un exemple de fichier de configuration :

LedgerName=KalimaLedger

NODE\_NAME=Node Client Example

NotariesList=62.171.131.154:9090,62.171.130.233:9090,62.171.131.157:9090,144.91.108.243:9090

FILES\_PATH=/home/rcs/jit/ClientExample

SerialId=PC1245Tuto

* LedgerName 🡺N’est pas encore utilisé dans la version actuelle
* NODE\_NAME 🡺 Vous pouvez mettre quelque chose qui permet de reconnaître votre nœud
* NotariesList 🡺 La liste des adresses et ports des notary, séparés par des virgules
* FILES\_PATH 🡺 C’est le chemin ou seront stockés les fichiers utiles à Kalima, ainsi que les logs
* serialId 🡺 C’est un identifiant qui va permettre l’autorisation sur la blockchain au premier lancement du node client (fournis par Kalima Systems dans le cas d’un essais sur nos Notary)

# Exécution du code

Pour tester votre projet, vous pouvez exécuter le code depuis Visual studio, ou depuis une console en ligne de commande. Il suffit de passer en paramètre, le chemin du fichier de configuration.

### Exécution depuis Visual Studio :

Dans Déboguer 🡺 Propriétés de débogage de … 🡺 Dans l’onglet Déboguer 🡺 Passer le chemin complet ou relatif du fichier de config dans « Arguments de l’application » (exemple : ../../../cfg/node.config). Enfin, appuyez sur F5 pour lancer l’application.

### Exécution en ligne de commande

Tout d’abord dans Visual studio, générez la solution en appuyant sur F6. Si la génération a réussi, l’exécutable et les dll nécessaires ont été copiés dans le dossier bin (dans bin/Debug/net48/ par exemple).

Ensuite depuis la console, dirigez vous dans le dossier de l’exécutable et lancer le en passant le fichier de configuration en paramètre :

cd Documents\Kalima\git\KalimaTuto\KalimaCSharpExample\KalimaCSharpExample\bin\Debug\net48

KalimaCSharpExample.exe ..\..\..\etc\cfg\node.config

# Résultats

Le programme d’exemple se connecte à la Blockchain, puis envoi 10 messages (1/seconde). Le TTL (Time To Live) de ces messages est de 10, ce qui signifie que chaque message sera automatiquement supprimé au bout de 10 secondes (une transaction aura lieu sur la blockchain pour chaque suppression). Ainsi, si votre code est correct, que vous avez correctement configuré le fichier de configuration, et que votre appareil est bien autorisé sur la blockchain, vous devriez avoir quelque chose de similaire dans votre console :

log\_srvMsg:KalimaMQ:KeyStore:60:setDevId deviceID=30dbd16c-1e0e-3265-8492-de8b14f9fb3e

log\_srvMsg:KalimaMQ:NioServer:60:NEW SERVER port ServerSocketChannel:9118

log\_srvMsg:KalimaMQ:Node:60:[connect new NioClient] 62.171.131.154:9090

log\_srvMsg:KalimaMQ:Node:60:[connect new NioClient] 62.171.130.233:9090

log\_srvMsg:KalimaMQ:Node:60:[connect new NioClient] 62.171.131.157:9090

log\_srvMsg:KalimaMQ:Node:60:[connect new NioClient] 144.91.108.243:9090

log\_srvMsg:NodeLib:Clone:60:onConnectedChange connected=true actual state=0

log\_srvMsg:KalimaMQ:Node:60:Node subscribe

log\_srvMsg:NodeLib:Clone:60:onConnectedChange connected=true actual state=0

log\_srvMsg:KalimaMQ:Node:60:Node subscribe

log\_srvMsg:NodeLib:Clone:60:onConnectedChange connected=true actual state=0

log\_srvMsg:KalimaMQ:Node:60:Node subscribe

log\_srvMsg:NodeLib:Clone:60:onConnectedChange connected=true actual state=0

log\_srvMsg:KalimaMQ:Node:60:Node subscribe

log\_srvMsg:KalimaMQ:Node:60:Node getSnapshotFromNotaryNodes cachePathsSeq= snapshotForAllCaches=true

log\_srvMsg:KalimaMQ:Node:60:Node getSnapshotFromNotaryNodes cachePathsSeq= snapshotForAllCaches=true

log\_srvMsg:NodeLib:Clone:60:addCache : /alarms/fire

log\_srvMsg:NodeLib:Clone:60:set received BEGINMEMCACHE sequence=2 previous=2 memCache.sequence=1

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/alarms/fire key=key sequence=2

log\_srvMsg:NodeLib:Clone:60:set ENDMEMCACHE cachePath=/alarms/fire HighestRemainingSequence=2 msg sequence=2 msg previous =2

log\_srvMsg:NodeLib:Clone:60:addCache : /sensors

log\_srvMsg:NodeLib:Clone:60:set received BEGINMEMCACHE sequence=383 previous=-1 memCache.sequence=-1

log\_srvMsg:NodeLib:Clone:60:set ENDMEMCACHE cachePath=/sensors HighestRemainingSequence=-1 msg sequence=383 msg previous =-1

log\_srvMsg:NodeLib:Clone:60:set received BEGINMEMCACHE sequence=2 previous=2 memCache.sequence=1

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/alarms/fire key=key sequence=2

log\_srvMsg:NodeLib:Clone:60:set ENDMEMCACHE cachePath=/alarms/fire HighestRemainingSequence=2 msg sequence=2 msg previous =2

log\_srvMsg:NodeLib:Clone:60:set received BEGINMEMCACHE sequence=383 previous=-1 memCache.sequence=383

log\_srvMsg:NodeLib:Clone:60:set ENDMEMCACHE cachePath=/sensors HighestRemainingSequence=-1 msg sequence=383 msg previous =-1

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key0 sequence=384

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key1 sequence=385

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key2 sequence=386

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key3 sequence=387

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key4 sequence=388

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key5 sequence=389

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key6 sequence=390

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key7 sequence=391

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key8 sequence=392

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key9 sequence=393

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key0 sequence=394

log\_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key0 seq=394 HighestRemainingSequence=393

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key1 sequence=395

log\_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key1 seq=395 HighestRemainingSequence=393

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key2 sequence=396

log\_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key2 seq=396 HighestRemainingSequence=393

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key3 sequence=397

log\_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key3 seq=397 HighestRemainingSequence=393

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key4 sequence=398

log\_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key4 seq=398 HighestRemainingSequence=393

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key5 sequence=399

log\_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key5 seq=399 HighestRemainingSequence=393

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key6 sequence=400

log\_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key6 seq=400 HighestRemainingSequence=393

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key7 sequence=401

log\_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key7 seq=401 HighestRemainingSequence=393

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key8 sequence=402

log\_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key8 seq=402 HighestRemainingSequence=393

log\_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key9 sequence=403

log\_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key9 seq=403 HighestRemainingSequence=393

Explication des résultats :

* Dans la première partie des logs, on peut voir que le programme se connecte à la blockchain, en se connectant en fait à chaque Notary.
* Ensuite une demande de snapshot est faite, ce qui permet à notre client de recevoir les données qu’il est autorisé à recevoir.
* Ensuite, le client va envoyer 10 messages en 10 secondes. Les messages seront reçus par tous les nodes autorisés sur la cache path en question, dont le vôtre. Ainsi, vous devez voir dans les logs une ligne pour chaque message envoyé (lignes commençant par  « StoreLocal »).
* Enfin, les messages seront supprimés un à un, puisque le TTL a été configuré sur 10 secondes. Vous devez donc voir les transactions dans les logs (lignes commençant par « StoreLocal remove »).