C# example :

1. Prerequisite

For this example, you must install an IDE for C#: Microsoft Visual Studio. Download link: https://visualstudio.microsoft.com

2. SerialId

The serialId is an identifier whose use will be explained in another paragraph of this document.

Note that to connect to Kalima blockchain it is necessary to use your own serialId communicated in the free trial email (insert it in the configuration file .config).

3. Execution of the KalimaCSharpExample project

To test the C# example, you can follow these steps:

Start Microsoft Visual Studio,

Click on the File menu ➔ Open ➔ Project/Solution ➔ Open the repository KalimaCSharpExample ➔ Click on the file KalimaCSharpExample.csproj ➔ Click on Open,

From Microsoft Visual Studio open the etc/cfg/node.config file ➔ Change the SerialId with the one you received in the free trial email,

From Microsoft Visual Studio open the Client.cs file,

Click on Debug menu ➔ Debug properties of ... ➔ In the tab Debug ➔ Put the full or relative path of the config file in "Application Arguments" (example : ../../../cfg/node.config). Finally, press F5 to launch the application.

4. Connection and interaction with Kalima blockchain

To connect and interact with the Kalima blockchain from your own project you can rely on the different explanations mentioned in the following of this document.

a. The creation of a Visual Studio project

In Visual Studio, create a new project and choose “Application console (.NET Core). Click on “Next”, then enter a name and choose a location for your project. Then, click on “Create”.

Afterwards, in « Solution Explorer », right click on your project → Edit the project file and replace the line :

```
<TargetFramework>netcoreapp3.1</TargetFramework>
```

By :

```
<TargetFramework>net48</TargetFramework>
```

b. Include the DLLs

On the right side, in the Solution Explorer, right click on your project → Add → New folder → Name it « libs ».

Copy-paste the DLLs present in the KalimaCSharpExample/libs in yours. Most of the DLLs come from ikvm :

https://github.com/jessielesbian/ikvm

IKVM is an implementation of Java for Microsoft .NET. If you need more Java features, please install ikvm and include the necessary DLLs in your project.

Then, in Visual Studio, right click on your project → Add → Project Reference → Browse → Select all the DLLs that you've addded in your libs folder.

c. KalimaClientCallBack

Create a new class under called « KalimaClientCallBack » :

- ⇨ Right click on your project
- ⇨ Add
- ⇨ New element
- ⇨ Select « class » and enter « KalimaClientCallBack.cs » in the field « name »
- ⇨ Add

Find below the complete code of the class (the namespace can change depending on the name of your project)

At this stage, the project includes errors which will be corrected later.

```
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_CON
NECTED) ? new AtomicBoolean(true) : new AtomicBoolean(false), nioClient,
false);
    }

    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) {}
      }
}
```

d. KalimaServerCallBack

In the same way, add a KalimaServerCallBack class. Find below the complete code:

```
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,
bool fromNotaryNode) { }

    public KMessage processNextMessage(KMessage msg, SocketChannel soc)
    {
      return null;
    }

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

e. Client

Lastly, rename your main class (by default Program.cs) by Client.cs → Right click on Program.cs → Rename.

Find below the complete code of the class :

```
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);
       }
     }
```

```
    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;
     }

   }
```

f. Configuration file

Add a cfg folder in your project, and place a « node.config » file in it. That configuration file will be passed as a parameter of the application.

See below for an example of a configuration file :

```
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 ➔Is not yet used in the current version
- NODE_NAME ➔ You can put something which allows you to recognize your node
- NotariesList ➔ The address and port list of the notary, separated by commas

- FILES_PATH ➔ It is the path where the files useful for Kalima will be stored, as well as the logs
- serialId ➔ It is an ID which will allow the authorization on the blockchain of the first launch of the client node (provided by Kalima Systems in the case of trials on our Notary)

g. Code execution

To test your project, you can execute the code from Visual Studio, or from an online command console. You need to pass the configuration file as parameter.

Execution from Visual Studio

In Debug ➔ Properties of debug of … ➔ In the Bebug tab ➔ Pass the complete path or relative of the configuration file in "Arguments of the application" (example : ../../../cfg/node.config). Lastly, click on F5 to launch the application.

Command-line execution

First, in Visual Studio, generate the solution by clicking on F6. If the generation works, the executable and the necessary DLLs have been copied in the bin folder (in bin/Debug/net48/ for example).

Then, from the console, go to the executable folder and launch it by passing the configuration file as parameter :

```
cd
Documents\Kalima\git\KalimaTuto\KalimaCSharpExample\KalimaCSharpExample\
bin\Debug\net48
KalimaCSharpExample.exe ..\..\..\etc\cfg\node.config
```

5. Results

The example program connects to the Blockchain, then sends 10 messages (1/second). The TTL (Time To Live) of those messages is of 10 which means that each message will be automatically deleted after 10 seconds (a transaction will happen on the blockchain for each deletion). Thus, if your code is correct, if you have configured the configuration file correctly, and if your device is authorized on the blockchain, you should have something similar to this in your 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] 167.86.124.188:9090

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:KalimaMQ:Node:60:[handleConnection add node ] 62.171.131.157:9090
myClients.size=2

log_srvMsg:KalimaMQ:Node:60:[handleConnection add node ] 144.91.108.243:9090
myClients.size=2

log_srvMsg:KalimaMQ:Node:60:[handleConnection add node ] 62.171.130.233:9090
myClients.size=4
```

```
log_srvMsg:KalimaMQ:Node:60:[handleConnection add node ] 167.86.124.188:9090
myClients.size=3

log_srvMsg:KalimaMQ:Node:60:Node subscribe

log_srvMsg:KalimaMQ:Node:60:Node subscribe

log_srvMsg:KalimaMQ:Node:60:Node subscribe

log_srvMsg:KalimaMQ:Node:60:Node subscribe

log_srvMsg:KalimaMQ:Node:60:Node getSnapshotFromNotaryNodes
snapshotForAllCaches=true

log_srvMsg:NodeLib:Clone:60:addCache : /alarms/fire

log_srvMsg:NodeLib:Clone:60:addCache : /alarms/fire.hdr

log_srvMsg:NodeLib:Clone:60:addCache : /alarms/fire.val

log_srvMsg:NodeLib:Clone:60:addCache : /alarms/fire.fmt

log_srvMsg:NodeLib:Clone:60:addCache : /alarms/fire.json

log_srvMsg:NodeLib:Clone:60:addCache : /sensors

log_srvMsg:NodeLib:Clone:60:addCache : /sensors.hdr

log_srvMsg:NodeLib:Clone:60:addCache : /sensors.val

log_srvMsg:NodeLib:Clone:60:addCache : /sensors.fmt

log_srvMsg:NodeLib:Clone:60:addCache : /sensors.json

log_srvMsg:NodeLib:Clone:60:addCache : /Kalima_Scripts

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/Kalima_Scripts key=Kalima-
Tuto/etc/scripts/reverse_string.js sequence=1

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/Kalima_Scripts
key=KalimaContractsTuto/KalimaExamples/reverse_string.js sequence=3

log_srvMsg:NodeLib:Clone:60:addCache : /Kalima_Scripts.hdr

log_srvMsg:NodeLib:Clone:60:addCache : /Kalima_Scripts.val

log_srvMsg:NodeLib:Clone:60:addCache : /Kalima_Scripts.fmt

log_srvMsg:NodeLib:Clone:60:addCache : /Kalima_Scripts.json

log_srvMsg:NodeLib:Clone:60:addCache : /Kalima_User

log_srvMsg:NodeLib:Clone:60:addCache : /Kalima_User.hdr

log_srvMsg:NodeLib:Clone:60:addCache : /Kalima_User.val

log_srvMsg:NodeLib:Clone:60:addCache : /Kalima_User.fmt

log_srvMsg:NodeLib:Clone:60:addCache : /Kalima_User.json

log_srvMsg:NodeLib:Clone:60:addCache : /Kalima_Password

log_srvMsg:NodeLib:Clone:60:addCache : /Kalima_Password.hdr

log_srvMsg:NodeLib:Clone:60:addCache : /Kalima_Password.val
```

```
log_srvMsg:NodeLib:Clone:60:addCache : /Kalima_Password.fmt

log_srvMsg:NodeLib:Clone:60:addCache : /Kalima_Password.json

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key0 sequence=623

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key1 sequence=624

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key2 sequence=625

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key3 sequence=626

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key4 sequence=627

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key5 sequence=628

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key6 sequence=629

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key7 sequence=630

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key8 sequence=631

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key9 sequence=632

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key0 sequence=633

log_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key0
seq=633 HighestRemainingSequence=632

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key1 sequence=634

log_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key1
seq=634 HighestRemainingSequence=632

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key2 sequence=635

log_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key2
seq=635 HighestRemainingSequence=632

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key3 sequence=636

log_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key3
seq=636 HighestRemainingSequence=632

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key4 sequence=637

log_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key4
seq=637 HighestRemainingSequence=632

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key5 sequence=638

log_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key5
seq=638 HighestRemainingSequence=632

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key6 sequence=639

log_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key6
seq=639 HighestRemainingSequence=632

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key7 sequence=640

log_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key7
seq=640 HighestRemainingSequence=632

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key8 sequence=641
```

```
log_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key8
seq=641 HighestRemainingSequence=632

log_srvMsg:NodeLib:MemCache:60:StoreLocal cachePath=/sensors key=key9 sequence=642

log_srvMsg:NodeLib:MemCache:60:StoreLocal remove cachePath= /sensors key=key9
seq=642 HighestRemainingSequence=632
```

Results explanation :

- In the first part of the logs, we can see that the program connects to the blockchain by connecting to each Notary.
- Then, a snapshot request is made, which allows our client to receive data it's been authorized to receive.
- Afterwards, the client will send 10 messages in 10 seconds. The messages will be received by all the nodes authorized on the cache path, including yours. Thus, you will have, in the logs, a line for each message sent (lines starting with « `StoreLocal` »).
- Lastly, the messages will be deleted one by one, since the TLL has been configured on 10 seconds. You will be able to see the transactions in the logs (lines starting with « `StoreLocal remove` »).