

Numeric Gold



SOMMAIRE

- 1 | [Administrer vos transactions avec NGold](#)
- 2 | [Quelques exemples des applications](#)
- 3 | [Point abordé dans ce manuel](#)
- 4 | [Partie 1 : INSTALLATION](#)
- 5 | [Plates-formes](#)
- 6 | [Compilateur](#)
- 7 | [Préparation de l'environnement de développement](#)
- 8 | [Installation de la bibliothèque NGold](#)
- 9 | [Conclusion](#)
- 10 | [Partie 2 : Développement et Publication](#)
- 11 | [IDE](#)
- 12 | [Header](#)
- 13 | [Compilation](#)
- 14 | [Conclusion](#)
- 15 | [Remerciements](#)

I. Partager

a. Administrez vos transactions avec NGOLD



Mise à jour : 13/12/2021

Durée d'étude : 5Ans.

Par AnetoEnterprise Inc. tout droit réservé.

Vous avez une agence de transfert d'argent ou une banque que vous voulez perfectionner vos transactions et attirer vos clients ?

La monnaie numérique ou de l'or numérique est à votre portée pour effectuer cela.

Ce manuel porte sur **NGold (Numeric Gold)**, un système des transactions monétaires opérationnel **24H/24** et **7J/7** sans connexion internet grâce à une bibliothèque statique pour permettre à tous les programmeurs dans le monde de l'adapter facilement en utilisant le compilateur **G++** pour la compilation de leurs applications des transactions monétaires. Le système **NGold** peut fonctionner sans connexion internet afin d'assurer la sécurité des établissements financiers et cela n'empêche pas de l'adapter aussi à une application web pour les services distants aux abonnés de votre établissement financier.

Le système peut être compilé afin de l'utiliser à d'autres langages des programmations tels que le **PHP** grâce au **CGI C++** et **JavaScript** par **NodeJS**.

b. Quelques exemples des applications :

Vous gérez une agence des transactions monétaire ou vous travaillez dans une banque et que vous voulez sécuriser vos transactions avec une technologie de chiffrement des informations avancées, **NGold** est mieux placé pour assurer leurs sécurités.

Vous voulez associé votre agence des transferts d'argent ou une banque à une autre sans pour autant exploiter vos données à la portée des hackers informatiques, la solution est à votre portée

tant que vos données seront sécurisées depuis votre ordinateur non connecté.

Vous voulez créer une application web distante afin de faciliter vos clients à effectuer les transactions partout où ils se trouvent sans se déplacer, **NGold** assure la sécurité d'un établissement financier sans connexion internet associé à une application web distante pour les clients.

Pour l'assurance et la sécurité des dépenses effectuées par le gouvernement, **NGold** assure la protection contre les détournements monétaires inutiles.

c. **Point abordé dans ce manuel :**

Avertissement :

La bibliothèque **NGold** est protégée par la loi relative au droit d'auteur et par les conventions internationales. Toute reproduction ou distribution partielle ou totale de ce logiciel sans autorisation de la société **AnetoEnterprise Inc.**, par quelque moyen que ce soit, est strictement interdite. Toute personne ne respectant pas ces dispositions se rendra coupable du délit de contrefaçon et sera passible des sanctions pénales prévues par la loi.

Seuls les codes sources présentés dans ce manuel qui sont **Open Source** (Sous licence GNU). C'est-à-dire peut être **copié, modifier et partager** gratuitement pour les établissements financiers tels que les **Banques, Agences des transferts d'argent et autres** qui veulent se lancer à leurs propres monnaies numériques fonctionnelle avec ou sans connexion internet.

En effet, il existe un système de gestion de monnaies numériques opérationnel sur internet nommé **Blockchain**. Tel est le système des monnaies numériques existantes du **Bitcoin, Ethereum et autres**. Comme vous le savez, sa technologie fonctionne de manière que toutes les transactions monétaires devraient être validées par les mineurs (Tel est le surnom des administrateurs de ce système de gestion **Blockchain**) enfin que les consommateurs ou les utilisateurs de ce système ne gardent pas une copie de la monnaie en cours d'utilisation ou déjà utilisée pour ne pas perturber le système ou ne pas utiliser une monnaie deux fois à des destinataires différents ou d'un seul destinataire.

Contrairement à ce système des transactions monétaires **NGold**, il fonctionne avec ou sans connexion internet afin d'assurer la sécurité premièrement des établissements financiers auquel utilisent sa bibliothèque et de protéger les informations relatives à la monnaie contre les hackers informatiques. Il exécute

ses traitements des transactions sans l'aide des mineurs ou administrateurs de la monnaie numérique même si les consommateurs ou les utilisateurs de la monnaie numérique sont connectés à internet ou hors connexion internet. Toutes les tâches ne s'effectuent automatiquement, aucun **Token**, pas de **Serveur Central**, non plus une **base des données** relatives aux consommateurs. Seule la monnaie circule avec ses propres informations.

C'est ce qui fait que la monnaie générée par le système **NGold** soit une monnaie à l'exécution automatique des données (Stand Alone Data). Il laisse le choix aux utilisateurs expéditeurs de spécifier l'endroit sur lequel leurs destinataires ou bénéficiaires des monnaies vont récupérer leurs monnaies en espèce avant les transactions et une fois les transactions effectuées, les bénéficiaires seront libres de récupérer leurs argent tant qu'ils seront les seuls à connaître les codes des validations de leurs monnaies et les endroits exactes pour les retraits. Tout cela sans connexion internet.

Aucun hacker ne sera en mesure de détecter les informations relatives à la monnaie transférée tant qu'elle sera chiffrée et aucun établissement financier ne sera connecté pour assurer la sécurité de toutes les parties participantes. Vous pouvez garder la copie de la monnaie comme bon vous semble pour tenter d'effectuer successivement deux retraits pour une monnaie valide, le système **NGold** a été conçu afin de bloquer toutes les requêtes forcées ou déjà utilisées tant que la monnaie générée sera quasiment utilisée ou validée dans un établissement financier spécifique.

Passons maintenant aux choses sérieuses.

II. Partie 1 : INSTALLATION

Dans ce chapitre, vous commencerez par apprendre les différentes plates-formes et compilateurs.

Introduction

Avant de pouvoir jouer avec les informations relatives à la monnaie, il vous faut connaître quelques plates-formes et compilateur que nous allons aborder ici.

a. Plates-formes

La bibliothèque **NGold** peut être installée dans les différentes plates-formes telles que : **Windows XP/7/8/10**, **Linux**, **Mac OSX**, **FreeBSD** et **Unix**. Parmi les plates-formes listées compatible pour l'intégration de la bibliothèque de chiffrement des informations monétaires **NGold**,

nous espérons que votre établissement financier aussi utilise l'une d'entre elles.

b. Compilateur

L'intégration de la bibliothèque NGold dans votre application de service, requise l'installation d'un compilateur **G++** afin de l'adapté non seulement au langage de programmation C++ mais à d'autres langage de programmation aussi.

1. Préparation de l'environnement de développement :

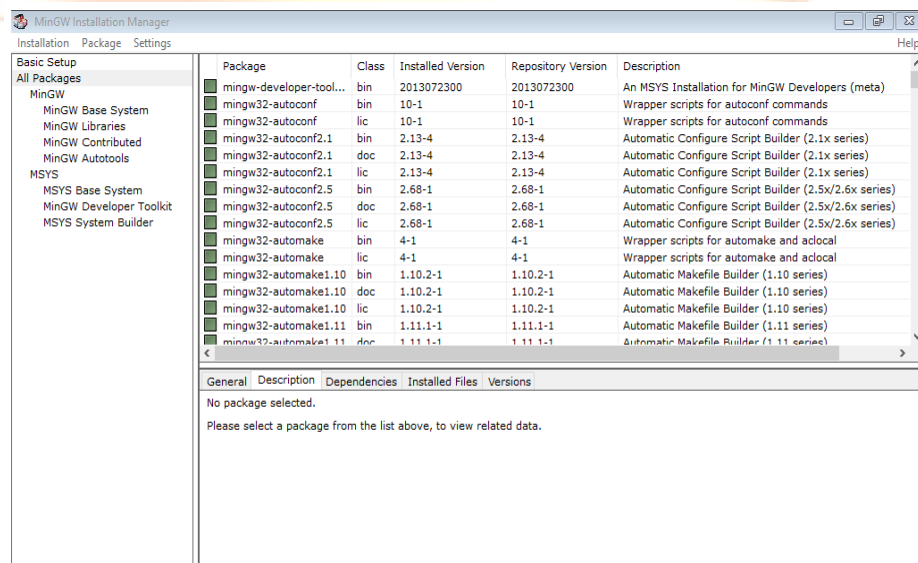
Nous allons maintenant préparer notre environnement de développement afin d'installer un compilateur G++ pour faciliter l'intégration de la bibliothèque NGold à votre application pour la gestion des monnaies numériques chiffrées. Malheureusement nous avons pu préparer les exemples concernant deux plates-formes.

✓ Windows :

Sous windows XP/7/8/10 vous avez deux choix pour préparer votre environnement de développement. Vous avez le choix

d'installer **MinGW**  ou **MSYS** .

Avec MinGW, une fois installée, vous avez le choix de cocher tous les composants ou de choisir ceux de votre choix afin de préparer votre environnement correctement comme sur l'image ci-après :

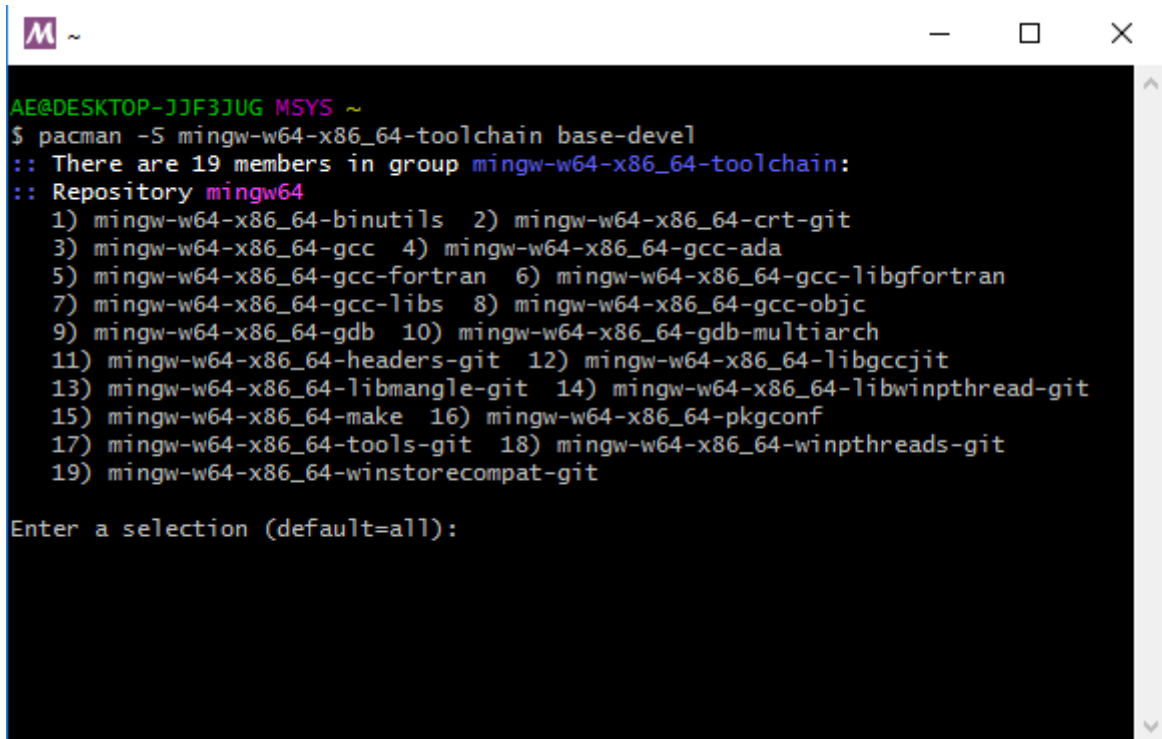


Dans notre cas, nous avons pu cocher le compilateur GCC et **G++** :

<input checked="" type="checkbox"/>	mingw32-gcc-g++	bin	6.3.0-1	6.3.0-1	The GNU C++ Compiler
<input checked="" type="checkbox"/>	mingw32-gcc-g++	dev	4.8.2	4.8.2	The GNU C++ Compiler

En utilisant MSYS en tant qu'environnement de développement, vous devez installer le compilateur **GCC** et **G++** en ligne de commande comme suite :

pacman -S mingw-w64-x86_64-toolchain base-devel



```
AE@DESKTOP-JJF3JUG MSYS ~  
$ pacman -S mingw-w64-x86_64-toolchain base-devel  
:: There are 19 members in group mingw-w64-x86_64-toolchain:  
:: Repository mingw64  
1) mingw-w64-x86_64-binutils 2) mingw-w64-x86_64-crt-git  
3) mingw-w64-x86_64-gcc 4) mingw-w64-x86_64-gcc-ada  
5) mingw-w64-x86_64-gcc-fortran 6) mingw-w64-x86_64-gcc-libgfortran  
7) mingw-w64-x86_64-gcc-libs 8) mingw-w64-x86_64-gcc-objc  
9) mingw-w64-x86_64-gdb 10) mingw-w64-x86_64-gdb-multiarch  
11) mingw-w64-x86_64-headers-git 12) mingw-w64-x86_64-libgccjit  
13) mingw-w64-x86_64-libmangle-git 14) mingw-w64-x86_64-libwinpthread-git  
15) mingw-w64-x86_64-make 16) mingw-w64-x86_64-pkgconf  
17) mingw-w64-x86_64-tools-git 18) mingw-w64-x86_64-winpthread-git  
19) mingw-w64-x86_64-winstorecompat-git  
  
Enter a selection (default=all):
```

✓ **Linux :**

Avec les distributions linux comme par exemple **Ubuntu** et **Debian**, vous pouvez installer le compilateur **GCC** et **G++** en utilisant la commande suivante depuis votre terminal :

sudo apt-get install gcc g++ -y

Pour la plate-forme linux **CentOS** :

sudo yum install gcc g++ -y

2. Installation de la bibliothèque NGold

Il est grand temps que jamais d'installer notre fameuse bibliothèque **NGold** afin de l'intégrer dans l'application en tant que bibliothèque statique des chiffrements des monnaies numériques. Les procédures d'installation sont similaires peu importe le système d'exploitation (Plate-forme) que vous utilisez.

Premièrement vous devez télécharger la bibliothèque NGold si c'est ne pas déjà fait et depuis votre terminal tapez la commande ci-après afin de la télécharger :

```
cd --
```

```
wget https://www.sourceforge.net/p/ngold/ngold-1.0.tar.xz
```

Une fois le paquet téléchargé dans votre disque dur, exécuter les commandes suivantes afin d'installer la bibliothèque **NGold** pour assurer son développement :

```
tar -xvf ngold-1.0.tar.xz
```

```
cd ngold-1.0/
```

```
./install.sh
```

3. Conclusion

Nous venons de préparé notre environnement de développement et installé complètement la bibliothèque NGold. Dans la deuxième partie, nous allons voir comment procéder pour appeler notre bibliothèque de chiffrement.

III. Partie 2 : Développement et Publication

Si vous êtes arrivé à ce point, sachez-le que vous êtes déjà dans le bon sens. Et vous devez savoir que l'intégration de la bibliothèque **NGold** ne requise pas de l'expérience du langage de programmation **C++**. Pour aller plus loin avec le langage de programmation **C++**, vous devez trouver sur internet les tutoriels qui sont basé sur sa programmation. Dans ce manuel, nous allons uniquement procéder à la programmation de l'intégration de la bibliothèque NGold.

a. IDE (Integrated Development Environment)

Un IDE est l'environnement de développement intégré conçu pour permettre aux développeurs informaticiens de codé leurs applications facilement avec des couleurs des syntaxes pour que les syntaxes soient lisibles à la programmation. Vous pouvez choisir n'importe quel éditeur de texte comme **Notepad** de Windows, **Notepad++**, **DevC++**



ou autres. Dans ce manuel, sous Windows nous allons utiliser le IDE DevC++ et sous Ubuntu, Notepad++.

b. Le Header

Le header comme son nom l'indique, c'est l'entête de notre bibliothèque des chiffrements monétaires. C'est dans ce fichier d'entête que contiendra toutes les fonctions concernant notre bibliothèque NGold afin de les adapter à votre application. Pour procéder, placez-vous dans le répertoire de votre projet et créez un fichier **ngold.h** qui sera notre Header et ajoutez-y les codes ci-après :

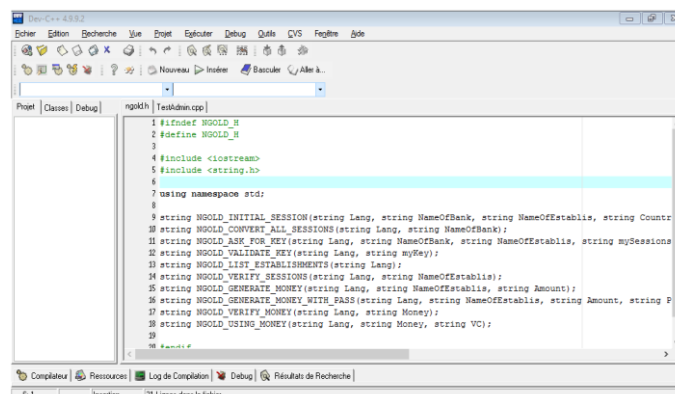
```
#ifndef NGOLD_H
#define NGOLD_H
```

```
#include <iostream>
#include <string.h>
```

```
using namespace std;
```

```
string NGOLD_INITIAL_SESSION(string Lang, string NameOfBank,
string NameOfEstablis, string Country, string Ville, string Commune,
string Quartier, string Rue, string RueNum, string Ref);
string NGOLD_CONVERT_ALL_SESSIONS(string Lang, string
NameOfBank);
string NGOLD_ASK_FOR_KEY(string Lang, string NameOfBank, string
NameOfEstablis, string mySessionsNumber);
string NGOLD_VALIDATE_KEY(string Lang, string myKey);
string NGOLD_LIST_ESTABLISHMENTS(string Lang);
string NGOLD_VERIFY_SESSIONS(string Lang, string NameOfEstablis);
string NGOLD_GENERATE_MONEY(string Lang, string
NameOfEstablis, string Amount);
string NGOLD_GENERATE_MONEY_WITH_PASS(string Lang, string
NameOfEstablis, string Amount, string Pass);
string NGOLD_VERIFY_MONEY(string Lang, string Money);
string NGOLD_USING_MONEY(string Lang, string Money, string VC);
```

```
#endif
```



Nous venons de définir notre Header, maintenant nous allons passer à la création d'une application d'exemple pour notre projet. Créons un autre fichier et donnons-leur le nom de **TestAdmin.cpp** comme sur l'image :



Exemple 1 :

Premièrement avant de commencer à code notre nouvelle application des transactions monétaires chiffrées, nous allons faire appel à notre fichier Header **ngold.h** dans l'entête de notre fichier **TestAdmin.cpp** comme suite :

```
#include "ngold.h"
#include <iostream>
#include <string>
using namespace std;
int main(int argc, char** argv) {

return EXIT_SUCCESS;
}
```

NB : L'intégration de la bibliothèque NGold dépend de ce que vous voulez faire. Vous n'êtes pas obligé d'utiliser toutes les fonctions définies dans le fichier Header. Ça dépend de votre idée et de service que vous voulez proposer à vos clients.

Mais la seule chose à savoir, si vous voulez concevoir une application concernant votre établissement financier, vous devez d'abord appeler la fonction d'initialisation de la session à votre application en utilisant la commande **NGOLD_INITIAL_SESSION**.

Cette commande permette d'initialiser la session de l'ordinateur sur lequel exécuteront les tâches des transactions monétaires. Un ordinateur qui n'est pas initialisé avec cette fonction ne sera pas en mesure de traiter les informations concernant les transactions chiffrées par la bibliothèque NGold.

Cette fonction **NGOLD_INITIAL_SESSION** utilise **10** paramètres afin de bien initialiser l'ordinateur concernant les transactions.

Pourquoi tous ces paramètres ?

Chacun d'entre eux à son utilité dès la première session de l'application d'administration dans votre ordinateur.

- ✓ **Lang** : Ce paramètre permet aux développeurs informaticiens de définir la langue sur laquelle la bibliothèque NGold va renvoyer la réponse en cas du succès ou d'échec de la requête. Pour cette première version, ce paramètre utilise deux valeurs. Le **en** pour les pays anglophone et le **fr** pour les pays francophone ;
- ✓ **NameOfBank** : C'est avec ce paramètre que l'établissement financier va définir le nom concernant son entreprise ;
- ✓ **NameOfEstablis** : Une fois définis correctement le nom de votre entreprise, vous serez censé de définir aussi le nom de l'ordinateur sur lequel exécutera les transactions. Par exemple, vous avez une banque et avec cette banque vous avez 5 établissements qui vont gérer les transactions. Alors, chacun de ces ordinateurs devrait avoir un nom unique pour assurer la sécurité de la monnaie chiffrée ;
- ✓ **Country** : Vous pouvez définir ici le pays sur lequel se trouve votre établissement (Votre ordinateur de gestion) ;
- ✓ **Ville** : La ville exacte de votre établissement ;
- ✓ **Commune** : La commune de votre établissement ;
- ✓ **Quartier** : Le quartier d'où vous êtes installé ;
- ✓ **Rue** : La rue de l'établissement ;
- ✓ **RueNum** : Le numero parcellaire ;
- ✓ **Ref** : Et une petite référence afin que vos clients arrivent a bien vous retrouver facilement grâce à ces informations que la bibliothèque NGold chiffrera pour initialiser votre ordinateur et

de le sécuriser contre les pirates informatique qui voudrions vous espionner pour une monnaie invalide ou envoyer une personne indésirable pour le retrait.

Pour se faire, ajoutons la fonction d'initialisation de la session à notre fichier **TestAdmin.cpp** comme suite :

```
#include "ngold.h"
#include <iostream>
#include <string>
using namespace std;
int main(int argc, char** argv) {
    string resultat="";
    resultat=NGOLD_INITIAL_SESSION("en", "MyBankA", "EstablishmentA", "DRC",
    "Kinshasa", "Ngaliema", "Ngoma Kinkusa", "Route de matadi", "200", "En
    diagonale avec Peloustore");
    cout << resultat << endl;
    return EXIT_SUCCESS;
}
```

Maintenant testons ensemble afin de voir le resultat de notre fonction d'initialisation. Avant de tester notre application d'essai, nous devons d'abord compiler tous nos deux fichiers **ngold.h** et **TestAdmin.cpp** afin qu'ils deviennent une application exécutable selon la plate-forme que nous utilisons pour le développement.

c. Compilation

Pour effectuer un appel de la bibliothèque NGold que nous avons installée autrement à notre compilateur G++, nous devons exécuter la commande suivante depuis un terminal **MinGW** ou **MSYS** :

```
cd E:/votredossier/exemple
g++ TestAdmin.cpp -o TestAdmin -lngold
./TestAdmin.exe
```

On voit que le **-lngold** fait partie de l'appel de notre bibliothèque NGold avant la compilation d'un fichier exécutable.

Et voilà le résultat :

```
root@AnetoEnterprise:/media/ae/AE/AE/AnetoEnterprise/GOLD/NOUVEAU/exemple# ./TestNGold
{"SUCCES":{"RESULT":"The key to your establishment is:","KEYS":"
-01111111--00000000--11111000--11111100--0111111--00001111--1111111111--11000000--10000000--0111111111--0000
000011--0000011111--0000000000--0000000111--0000000001--0000011111--11111100--11000000--0000000000--00000
00000--0000000000--0000000111--11000000--11100000--0011111--0000000001--0000011111--111111--11110000--000
0011--000011--01111111--11111100--11000000--1111111111--0000000111--1111111111--01111111--00000000--1111100
0--11111100--01111111--00001111--"}}}
```

Cette clé générée est gratuite avec **1000 sessions** pour vous permettre de tester correctement votre application pour les monnaies des tests version **SandBox**. Pour la version **Production** et **Commerciale**, nous allons voir à la fin de ce manuel comment procéder pour avoir la clé d'une vraie monnaie numérique chiffrée. Et vous pouvez voir depuis l'emplacement de votre fichier exécutable que vous venez de compilée et exécutée, un fichier avec l'extension **.ng** portant le nom de votre établissement (Ordinateur) :



Establishm
entA.ng

Dans notre exemple nous avons pu définir comme nom d'ordinateur que nous utilisons à **EstablishmentA**. C'est pour cela que ce fichier aussi porte ce nom d'initialisation y compris l'extension **ng**.

Si votre ordinateur n'est pas compatible pour l'utilisation de la bibliothèque NGold, vous aurez la réponse d'échec comme suite :

```
/e/AE/AnetoEnterprise/GOLD/NOUVEAU/exemple
AE@DESKTOP-JJF3JJUG MSYS /e/AE/AnetoEnterprise/GOLD/NOUVEAU/lib
$ cd E:/AE/AnetoEnterprise/GOLD/NOUVEAU/exemple
g++ TestAdmin.cpp -o TestAdmin -lngold
./TestAdmin.exe
/bin/sh: line 1: lsblk: command not found
{"ERROR":{"RESULT":"Sorry, the operating system you are using is not compatible with this library."}}
AE@DESKTOP-JJF3JJUG MSYS /e/AE/AnetoEnterprise/GOLD/NOUVEAU/exemple
$
```

Ce message d'erreur **lsblk : command not found** défini que la bibliothèque NGold ne trouve pas votre disque dur avec cette commande. Alors, vous serez censé de l'installer afin que le système des chiffrements fonctionne correctement depuis votre ordinateur sur lequel exécutera toutes ses fonctions.

Nous voulons maintenant convertir notre clé générée des sessions gratuite afin qu'elle soit partagée à nos banques partenaires. Par exemple, vous travaillé en collaboration avec une autre banque ou agence des transferts d'argent et que vous voulez vos différents clients transfèrent l'argent sans pour autant changer de banque ou de l'agence, vous serez censé intégrer la fonction **NGOLD_CONVERT_ALL_SESSIONS**. Cette fonction permettra à NGold de convertir tous vos établissements utilisant l'extension **.ng** à un fichier portant l'extension **ngp**.

Comme nous avons précisé ci-dessus, si vous avez 5 établissements ou plus, à chacun initialisera son ordinateur et cette fonction sera exécuter en utilisant un ordinateur spécifique. Dans ce cas vous serez censé de récupérer tous les fichiers portant l'extension **.ng** dans chaque ordinateur initialisé afin de les convertir aisément à un fichier de partenariat chiffré.

Exemple 2 :

Créons une autre application, vous pouvez choisir n'importe quel nom et ajoutez-y les syntaxes ci-après :

```
#include "ngold.h"
#include <iostream>
#include <string>
using namespace std;
int main(int argc, char** argv) {
    string resultat="";
    resultat=NGOLD_CONVERT_ALL_SESSIONS("en", "MyBankA");
    cout << resultat << endl;
    return EXIT_SUCCESS;
}
```


Le paramètre **MyBankA** représente le nom de notre banque d'essai. Alors, à la fin de la conversion de tous les fichiers d'initialisation, le fichier de partenariat portera le nom de **MyBankA.ngp** afin qu'il soit partagé dans tous les ordinateurs de votre banque et des ceux de vos partenaires.

NB : Tous les fichiers portant l'extension .ng doivent se retrouver sur le même emplacement de votre application exécutable de conversion.

Maintenant vous pouvez compiler et tester votre nouvelle application en plaçant tous les fichiers portant l'extension .ng sur le même répertoire, vous verrez un nouveau fichier créé portant le nom de votre banque ou agence de transfert avec l'extension .ngp de partenariat chiffré.

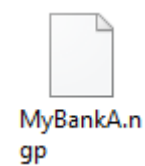
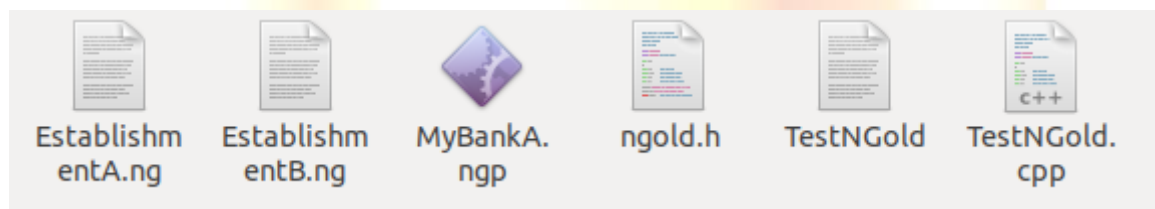
Sous Windows :

```
{"SUCCES":{"RESULT":"The establishment EstablishmentA, added correctly. The establishment EstablishmentB, added correctly. "}}
```



Sous Linux :

```
root@AnetoEnterprise:/media/ae/AE/AE/AnetoEnterprise/GOLD/NOUVEAU/exemple# ./TestNGold  
{"SUCCES":{"RESULT":"The establishment EstablishmentA, added correctly. The establishment EstablishmentB, added correctly. "}}
```



Vous êtes libre de partager ce fichier via tous les ordinateurs de votre établissement et vos partenaires aussi de leurs parts, partageront le fichier dans les ordinateurs qu'ils utilisent pour la gestion de monnaie numérique NGold en toute sécurité.

Exemple 3

Maintenant il est temps de créer une application cliente permettant à nos clients de transférer l'argent aisément dans toutes les plates-formes des communications ou dans tous les réseaux distants sans fil. Avec cette partie, vous avez le choix de choisir si votre application sera à la portée des utilisateurs distants connectés ou les utilisateurs distant non connectés.

A noter que, les utilisateurs non connectés sont les banques ou les agences partenaires sur lesquelles vous travaillez avec elles. Tandis que les utilisateurs non connectés seront vos clients qui veulent transférer de l'argent à leurs familles, camarades ou autres sans pour autant se déplacer ou utiliser une Carte Bancaire mais juste avec leurs téléphones mobile ou ordinateurs.

Avant de procéder, vous devez savoir que la génération de la monnaie numérique chiffrée requiert les deux fonctions **NGOLD_LIST_ESTABLISHMENTS** et **NGOLD_GENERATE_MONEY**.

- ✓ **NGOLD_LIST_ESTABLISHMENTS** : Cette fonction utilise un seul paramètre pour la langue de la réponse d'une requête. Elle permet de lister les établissements depuis un fichier de partenariat **.ngp** afin de récupérer toutes les informations concernant les endroits exactes de chaque établissement pour que les utilisateurs arrivent à bien choisir l'établissement sur lequel le bénéficiaire se déplacera pour la récupération de son argent.

NB : C'est un secret entre l'expéditeur et le bénéficiaire de la monnaie numérique générée.

- ✓ **NGOLD_GENERATE_MONEY** : Celui-ci utilise 3 ou 4 paramètres :
 - a) **Lang** : Comme d'habitude la langue de la réponse des résultats ;
 - b) **NameOfEstablis** : Le nom de l'établissement générateur de la monnaie (Dans notre exemple **EstablishmentA**) ;
 - c) **Amount** : Le montant à transférer au bénéficiaire ;
 - d) **Pass** : Et le mot de passe pour permettre au bénéficiaire de la monnaie de le définir avant son retrait. Ce paramètre n'est pas obligatoire mais c'est juste pour la sécurité des informations monétaires. Si vous ne le définissez pas, la bibliothèque **NGold** générera automatiquement un code de validation à votre place.

Passons à l'application, créer un autre fichier portant le nom de **TestClient.cpp** et ajoutez-y les syntaxes ci-après :

```
#include "ngold.h"

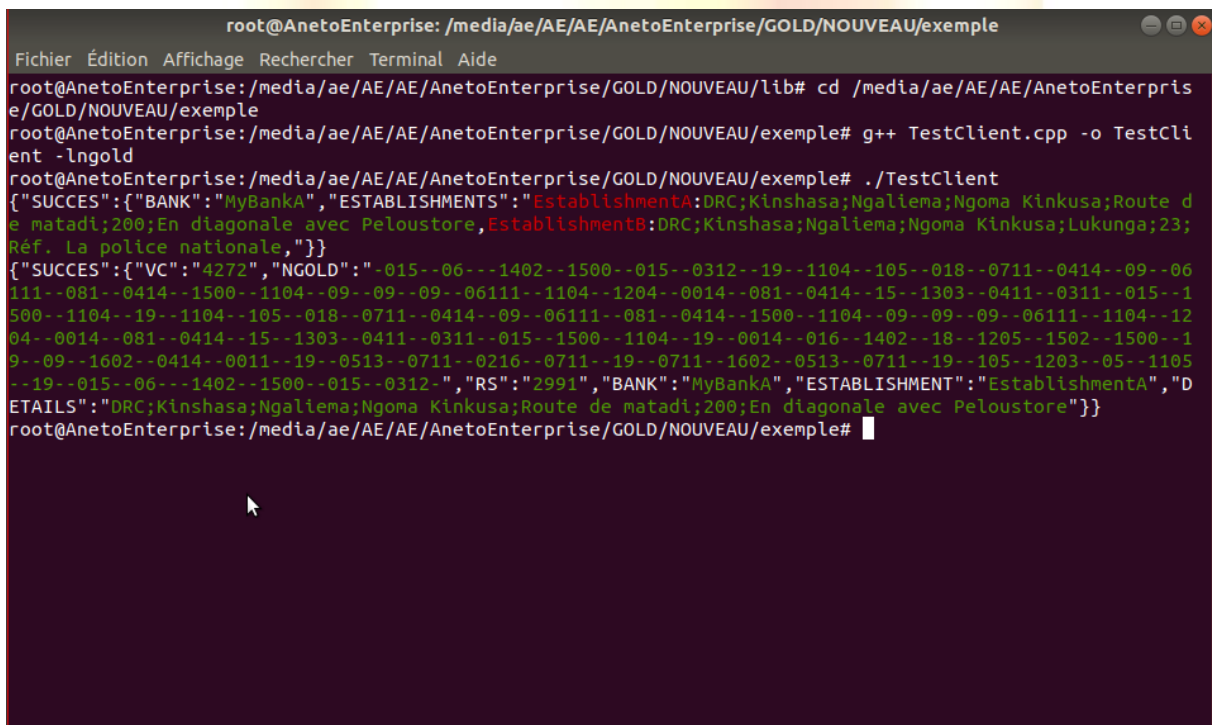
#include <iostream>

#include <string>

using namespace std;

int main(int argc, char** argv) {
    string resultat="";
    resultat=NGOLD_LIST_ESTABLISHMENTS("en");
    cout << resultat << endl;
    resultat=NGOLD_GENERATE_MONEY("en", "EstablishmentA", "0.5$");
    cout << resultat << endl;
    return EXIT_SUCCESS;
}
```

Compiler et exécuter l'application vous verrez votre monnaie numérique chiffrée correctement comme représenté sur l'image ci-dessous :



```
root@AnetoEnterprise: /media/ae/AE/AE/AnetoEnterprise/GOLD/NOUVEAU/exemple
Fichier Édition Affichage Rechercher Terminal Aide
root@AnetoEnterprise: /media/ae/AE/AE/AnetoEnterprise/GOLD/NOUVEAU/lib# cd /media/ae/AE/AE/AnetoEnterprise/GOLD/NOUVEAU/exemple
root@AnetoEnterprise: /media/ae/AE/AE/AnetoEnterprise/GOLD/NOUVEAU/exemple# g++ TestClient.cpp -o TestClient -lngold
root@AnetoEnterprise: /media/ae/AE/AE/AnetoEnterprise/GOLD/NOUVEAU/exemple# ./TestClient
{"SUCCES":{"BANK":"MyBankA", "ESTABLISHMENTS":{"EstablishmentA:DRC;Kinshasa;Ngaliema;Ngoma Kinkusa;Route de matadi;200;En diagonale avec Peloustore, EstablishmentB:DRC;Kinshasa;Ngaliema;Ngoma Kinkusa;Lukunga;23; Réf. La police nationale,"}}
{"SUCCES":{"VC":"4272", "NGOLD":"-015--06--1402--1500--015--0312--19--1104--105--018--0711--0414--09--06111--081--0414--1500--1104--09--09--09--06111--1104--1204--0014--081--0414--15--1303--0411--0311--015--1500--1104--19--1104--105--018--0711--0414--09--06111--081--0414--1500--1104--09--09--09--06111--1104--1204--0014--081--0414--15--1303--0411--0311--015--1500--1104--19--0014--016--1402--18--1205--1502--1500--19--09--1602--0414--0011--19--0513--0711--0216--0711--19--0711--1602--0513--0711--19--105--1203--05--1105--19--015--06--1402--1500--015--0312--", "RS":"2991", "BANK":"MyBankA", "ESTABLISHMENT":"EstablishmentA", "DETAILS":{"DRC;Kinshasa;Ngaliema;Ngoma Kinkusa;Route de matadi;200;En diagonale avec Peloustore"}}}
root@AnetoEnterprise: /media/ae/AE/AE/AnetoEnterprise/GOLD/NOUVEAU/exemple#
```

On voit clairement depuis l'image, l'application avait listé premièrement les établissements afin que l'utilisateur choisisse l'emplacement sur lequel le bénéficiaire récupérera son argent. Par la suite, la génération de la monnaie effectuée avec les informations suivantes :

- ✓ **VC** : Le VC est le code de validation (En anglais Validating Code) que le bénéficiaire utilisera pour récupérer son argent afin d'affirmer qu'il est le propriétaire de la monnaie ;
- ✓ **NGOLD** : Ce paramètre représente la monnaie numérique chiffrée que l'expéditeur copiera afin de l'envoyer au bénéficiaire comme monnaie valide ;
- ✓ **RS** : Comme son nom l'indique Rest Sessions (Sessions restantes), affiche le résultat des sessions restantes pour les prochaines générations des monnaies afin de savoir s'il vous reste combien des sessions valides tant que la bibliothèque NGold est payante pour chaque session de la génération de monnaie numérique chiffrée ;
- ✓ **BANK** : Affiche la banque ou l'agence destinatrice de la monnaie ;
- ✓ **ESTABLISHMENT** : L'établissement destinatrice de la monnaie ;
- ✓ **DETAILS** : Et les détails concernant l'établissement sur lequel le bénéficiaire se déplacera pour la récupération de son argent.

Nous venons de voir comment notre bibliothèque affiche le nombre des sessions restantes après la génération de la monnaie chiffrée. Alors, nous allons voir comment vérifier les sessions restantes sans pour autant générer la monnaie. La fonction qui nous donne cette possibilité est la **NGOLD_VERIFY_SESSIONS**. Cette fonction utilise deux paramètres **Lang** et **NameOfEstablis** afin de bien afficher les sessions restantes de votre ordinateur concernant la génération de la monnaie numérique chiffrée de NGold.

Exemple 4

Dans votre application existante ou fichier **TestAdmin.cpp** existant, ajouter les syntaxes ci-après :

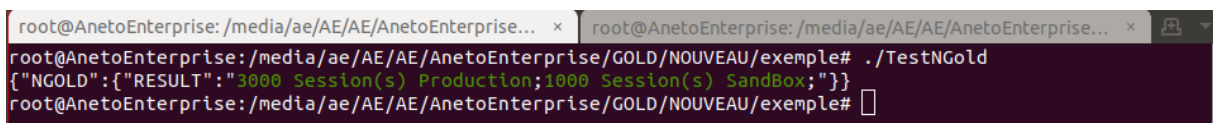
```
#include "ngold.h"
#include <iostream>
#include <string>
using namespace std;
int main(int argc, char** argv) {
```

```

string resultat="";
resultat=NGOLD_VERIFY_SESSIONS("en", "EstablishmentA");
cout << resultat << endl;
return EXIT_SUCCESS;
}

```

Compiler et exécuter l'application vous verrez les restes de vos sessions comme représenté ici :



```

root@AnetoEnterprise: /media/ae/AE/AE/AnetoEnterprise... x root@AnetoEnterprise: /media/ae/AE/AE/AnetoEnterprise... x
root@AnetoEnterprise: /media/ae/AE/AE/AnetoEnterprise/GOLD/NOUVEAU/exemple# ./TestNGold
{"NGOLD":{"RESULT":"3000 Session(s) Production;1000 Session(s) SandBox;"}}
root@AnetoEnterprise: /media/ae/AE/AE/AnetoEnterprise/GOLD/NOUVEAU/exemple# 

```

Il est maintenant grand temps que jamais pour que la banque ou l'agence partenaire récupère et vérifie la monnaie valide avant un retrait de son bénéficiaire. La fonction permettant de vérifier la monnaie valide est belle et bien **NGOLD_VERIFY_MONEY**. Elle est gavée des deux paramètres **Lang** et **Money** :

- ✓ **Lang** : La langue de la réponse pour la requête de vérification ;
- ✓ **Money** : Et la monnaie numérique chiffrée du bénéficiaire.

Exemple 5

On continue toujours avec notre fichier **TestAdmin.cpp**, ajouter les syntaxes ci-dessous pour la vérification de la monnaie valide ou non valide de vos clients :

```

#include "ngold.h"
#include <iostream>
#include <string>
using namespace std;
int main(int argc, char** argv) {
string resultat="";
resultat=NGOLD_VERIFY_MONEY ("en", "-015--06---1402--1500-");
cout << resultat << endl;
return EXIT_SUCCESS;
}

```

Afin de compilé et testé, nous avons le resultat ci-après contenant le nom de la banque ou agence de transfert, la valeur de la monnaie et son Status de validation :

```
root@AnetoEnterprise:/media/ae/AE/AE/AnetoEnterprise/GOLD/NOUVEAU/exemple# ./TestClient  
{"SUCCES":{"BANK":"MyBankA", "AMOUNT":"0.5$", "STATUS":"Production"}}
```

NB : Si la monnaie vérifiée est la monnaie de teste, votre Status sera de **Sandbox**. En outre, votre Status sera comme représenté sur l'image pour la **Production** (La vraie monnaie).

Exemple 6

Passons maintenant au retrait de la monnaie. Une fois que le bénéficiaire aura toutes les informations relatives à la récupération de son argent, il sera libre de passer à la banque ou agence de transfert d'argent destinée pour cette monnaie numérique afin de récupérer son argent aisément.

La fonction compatible pour effectuer un retrait est la **NGOLD_USING_MONEY** utilisant 3 paramètres **Lang**, **Money** et **VC** :

- ✓ **Lang** : Défini la langue de la réponse concernant la requête ;
- ✓ **Money** : Représente la monnaie à utiliser ;
- ✓ **VC** : Et le code de validation de la monnaie sur laquelle il a été généré.

Ajoutons les syntaxes suivantes au fichier **TestAdmin.cpp**, compilons et testons :

```
#include "ngold.h"  
#include <iostream>  
#include <string>  
using namespace std;  
int main(int argc, char** argv) {  
    string resultat="";  
    resultat=NGOLD_USING_MONEY("en", "-015--06---1402--1500-", "47135");  
    cout << resultat << endl;  
    return EXIT_SUCCESS;  
}
```


Comme resultat nous avons :

```
root@AnetoEnterprise:/media/ae/AE/AE/AnetoEnterprise/GOLD/NOUVEAU/exemple# ./TestAdmin
{"NGOLD":{"RESULT":"2988 Session(s) Production;1000 Session(s) SandBox;"}}
{"SUCCES":{"BANK":"MyBankA", "AMOUNT":"0.5$", "STATUS":"Production"}}
{"SUCCES":{"RESULT":"Validated !!"}}
```

Qui veut dire que notre retrait à était effectué correctement.

Nous sommes arrivés à la fin de nos exercices de tests, maintenant nous pouvons demander l'autorisation à la bibliothèque NGold de passer en mode Production pour présenter le produit final à nos consommateurs. Les deux fonctions qui nous restent sont la **NGOLD_ASK_FOR_KEY** et **NGOLD_VALIDATE_KEY**.

- ✓ **NGOLD_ASK_FOR_KEY** : Cette fonction requise 4 paramètres suivants et effectue une demande de la clé version **Production** :
 - **Lang** : Défini la langue de la réponse ;
 - **NameOfBank** : Le nom de la banque ou agence de transfert ;
 - **NameOfEstablis** : Le nom de l'établissement initiateur de la demande de la clé de production ;
 - **mySessionsNumber** : Et le nombre des sessions demandées.
- ✓ **NGOLD_VALIDATE_KEY** : Pour la validation de la clé dans votre ordinateur une fois activée depuis le support AnetoEnterprise, cette fonction requise 2 paramètres **Lang** et **myKey**.
 - **Lang** : Une fois de plus la langue de la réponse obtenue ;
 - **myKey** : Et la clé récupérée depuis le support de la société distributeur de la bibliothèque NGold.

Finissons-en !

Dans le même fichier **TestAdmin.cpp**, ajouter les syntaxes suivantes afin de lancer la demande et d'activer vos sessions :

```
#include "ngold.h"
#include <iostream>
#include <string>
using namespace std;
int main(int argc, char** argv) {
string resultat="";
resultat=NGOLD_ASK_FOR_KEY("en", "MyBankA", "EstablishmentA", "3000");
cout << resultat << endl;
```

```
resultat=NGOLD_VALIDATE_KEY("en", "-015--06---1402--1500-", "47135");  
cout << resultat << endl;  
return EXIT_SUCCESS;  
}
```

Compilons et testons l'application finale.

d. Conclusion

Nous venons de voir comment la bibliothèque NGold effectue ses traitement et renvoi les résultats au format XML afin de vous permettre de bien l'adapté à d'autres langages de programmations compatible à la lecture des informations au format XML. Nous espérons qu'après cette lecture vous serez capable d'innover votre établissement financier pour faciliter les transactions du futur à l'adaptation de votre interface graphique de l'application des services via la bibliothèque NGold.

IV. Remerciements

- Nous remercions premièrement notre **Dieu** enfin de nous avoir gardé en bonne santé durant toutes ces années des tests concernant cette bibliothèque NGold ;
- **Rodriguez Lufungula** pour son soutien moral et paix à son âme ;
- **Nagui Divangi** pour la lecture et validation de ce projet ;
- Et à **AnetoEnterprise Inc.** Pour le financement de ce projet.

ⁱ Copyright 2021 by AnetoEnterprise Inc. all rights reserved.