MARMARA CREDIT LOOPS

MARMARA CHAIN v.1.2.0

MARMARA CONNECTOR v.0.4.2 Alpha

MCL DEVELOPER GUIDE

2024

Contents

Contents	2
Marmara Credit Loops	3
What is MCL? What are the Features?	7
How Does MCL Work? How Can You Earn MCL?	9
Easy Installation of Marmara Blockchain and MCL Coin Production with Marmara Connector	10
Build Marmara from Source	14
Getting Started with Marmara	19
How to make Marmara Credit Loops	31
Installing Marmara from Precompiled Binaries	37
Understanding Marmara Chain	40
Unlock the Power of Marmara Blockchain Staking	41
Experience the Power of MCL Staking on Marmara Chain	43
Why Marmara (MCL) Coin Stands Out	45
Marmara Connector	47
MCL Concise Beginner's Guide	61
Transactions on Cryptocurrency Exchanges and Obtaining MCL	62
Answers to Frequently Asked Questions	65
Links	69
Important Notice to Users	71
License Details	72

Note: This document was prepared by Marmara community volunteers, using the official documentation of the Marmara Chain project and the information on the Marmara Discord channel of the community. Each source is mentioned under the relevant topic.

Marmara Credit Loops

MCL, the First and the only DeFi system in the World designed to run in Real Economy.

Marmara Credit Loops on MarmaraChain, a trust based/self assured peer-to-peer credit creation and circulation system, is the very first digitized version of the widely used analog post-dated cheques in the World. Marmara Credit Loops (MCL) have most of the features provided by the Komodo Blockchain technologies and have been developed based on the way of smart contracts with Custom Consensus (CC) system in Komodo Antara Framework.

In addition to being the very first digital version of the credit loops, due to its structure and conforming operations with the blockchain technology, Marmara Chain is very unique in the world that implements blockchain in such a way.

The credit loops are applied directly onto the blockchain core. It is not possible to intervene, manipulate or change it in any ways.

The properties of the system are summarized below:

Works best in Real Economy: MCL is a DeFi system designed to run in real economy during shopping. Both issuer and holders can earn from 3x staking during shopping in life cycle of a credit loop. Credits in a loop can be endorsed and circulated until maturity date.

Half of blocks comes as activated/locked: 50% of blocks found are already activated/locked. Hence, they readily join in staking process. You can unlock them anytime for spending or lock in credit loops for 3x staking power.

3x staking in Credit Loops: 3x staking when Activating/locking Coins in Credit Loops. They are settled automatically on maturity date and cannot be opened by anyone. But credits may be endorsed to buy goods and services. They act as money during shopping.

Nonmanipulable, nonintervenable credit loops: Credit loops cannot be stopped and credits in loops can circulate during shopping until maturity date. The nodes in a loop is maximum 1000.

Block Rewards: Block time is 1 minute and rewards per block are 30 Marmara (MCL).

75% Staking: Staking is only possible with activated/locked coins. You have the 3x staking power when staking with coins locked in Credit Loops.

25% Mining (PoW): MCL uses Equihash algorithm for Mining with PoW which has only 25% effect in block rewards. You can mine Marmara similar to Komodo and Zcash mining.

To the contrary belief, banks are not the sole institutions for creating money based on credit. In many countries all over the world, like banks, individuals are able to create money through post-dated cheques or promissory notes in a peer to peer manner. They work best in real economy.

Inspired by the paper-based blockchain nature of these promissory notes and post-dated cheques' circulation, in the same analogy, Marmara Credit Loops Project aims to digitalize those credits on the blockchain while solving the problem of non-redemption of these tools.

The early release of this Project, named as Version 1 (Protocol 1), solves the problem of non-redemption with 100% collateralization through trustless blockchain mechanism.

The second Protocol of the Project aims to solve the non-redemption problem based on a trust based blockchain system with the aid of a protective layered architecture of nodes in credit loop layer formed against defaulting.

Protocol 1 Millennium Version:
100% Collateralization Mode
Boosted Reinforcement of Purchasing Power
Issuers and Holders can earn with 3x staking while shopping with Credit Loops.

Protocol 2 Trust Based:
Zero Collateralization
Money Creation via Communities
People can create credits via their escrows in their communities.

Marmara v.1.0

100% Self-assured Credit Loops System

In Protocol 1, all credits are fully collateralized by issuers of credits. Therefore, there is no defaulting risk. For providing this, issuers in credit loops have the right for 3x staking until maturity dates of a credits.

Boosting Purchasing Power

Due to 100% assurance for redemption by issuers, there is no need for an escrow or aval at any point of shopping with credit loops in Protocol 1. Holders, i.e. providers of goods and services in a loop can benefit from 3x staking until they become endorsers by purchasing goods or services from new holders. By this way, they become endorsers and transfer their 3x staking right to new holders. There is no meaning for an endorser to transfer credit unless the transfer is for shopping since he/she transfer the 3x staking power.

MCL is the only staking system in the World where locked coins allow owners staking while they are circulating among people.

Estimation of Staking Rewards

One may estimate the staking rewards by using the formula below. Please note that the formula gives only estimated results which may be varying with parameters such as the number and quantity of both activated and locked-in loop coins, their aging, the internet connection, hard disk speed, number of cores etc. However, the minimum requirement for hardware is low with 2 cores, 4 GB RAM, SSD Harddisk and a reasonable internet connection. You can even stake with a miniPC easily.

$$\left(\frac{\frac{Activated}{TotalActivated} + \frac{Loops}{Locked\ in\ loops} x 3}{4} \right) x\, 30\, x\, 60\, x\, 24\, x\, 0.75$$

Activated: Activated amount (in your wallet)

Loops: Locked in loops (in your wallet)

*TotalActivated: Total activated amount

**Locked in Loops: Total Locked in loops amount

30: MCL per minute

60: Approx number of MCL blocks created per hour

24: 24 hours

0.75: Total staking rate

*,** : For Total Activated and Total Locked in loops amounts, visit here.

Marmara v.2.0

Trust Based Credit Loops

Escrow system and escrow services

Solution to Non-redemption in Trust Based Credit Loops

The requirement definitions have been completed for trust-based blockchain solution (Version 2) but not yet been fully implemented on the existing blockchain. The trust-based system works similarly to the existing trust-based analog blockchain in paper-based check checks, but with several preventive programs (3-layer recommendation), self-assurance is provided against the problem of non-payment (defaulting) or bouncing. In the trust-based blockchain system, the issuer creates a credit similar to the banking system. The difference here is that the credit created in blockchain is circulated in a credit loop, in contrast to the bank credit where the credit generally remains as the contract between the bank and the borrower. Non-redemption problem in trust based credit loops is solved in a unique way.

There are several layers of protection against defaulting, i.e. non-redemption. First of all, the issuance and settlement processes of trust based credit loops will be through Escrows acting as responsibility and trust centers. They can have additional guarantees to be used in case of no refunds.

Issuers and endorsers in the credit loop layer will most likely have active/locked coins to be used as a collaterals in case of defaulting. Avalists' activated funds in community layer may also be used collaterals.

The blockchain fund layer will also be used as a last resort against nonredemption.

Escrow System

In trust-based mode, the issuance or settlement of credit loops are performed by trusted escrow service providers, which act as distributed notaries. Those who provide this intermediary services are called "Escrows". All issuer nodes in credit loops must verify their data by an escrow service to join the credit loop system. The Escrow system is similar to Notarization system with more responsibilities.

Escrow Services
Mining/staking people can easily use the system with trustless features of Marmara Credit Loops. Data Management is only necessary for necessary people such as credit issuers and for trust based version.

Escrows and avalists in community layer provide an additional layer of trust in the blockchain. Currently, in the analog version of post-dated cheques, cheques are initiated by banks and banks do not issue checks to everyone. In case of promissory notes, avalists are more important. Escrows bear the main responsibility during issuance and settlement of peer-to-peer credits on Marmara system.

Data association for issuers must be through escrows. Escrows may be Web-based escrow services, Crypto Exchanges, Notaries, Post Offices, Law Offices, Credit Unions and even Banks. Not everyone can become an Escrow. Please contact us for details if you want to be an escrow.

To ensure that there is no identity theft in version 2, user (issuer) data should be managed by only through Escrows. This will be done by the Escrows with the their own Know Your Customer procedures, financial credit scoring system etc. Escrows will be responsible during issuance and settlement of trust based credit loops. Therefore, they will have earnings from blockchain fund in third layer planned for Version 2.

****
Source: https://marmara.io/marmaracreditloops#marmara_version

What is MCL? What are the Features?

MCL is a native blockchain, native cryptocurrency project inspired by the work of Post-dated Checks and Promissory Notes. For those who don't know, let me tell you. Although the check books for Post-dated Checks are issued by banks, it is an interest-free credit system that is an alternative to the banking system. Tradesmen and businessmen run an alternative credit system in the economy by issuing credit with post-dated checks while selling or purchasing goods and services.

THE PROBLEM OF BOUNCING CHECKS AND OTHER PROBLEMS

Although postdated checks and promissory notes are good instruments, their biggest problem is the bouncing, or non-redemption. This system, which is based on trust, collapses on top of each other with the domino effect, especially in times of economic crisis. MCL is a project developed to solve the non-repayment problem in such credits.

MCL MAIN FEATURES

MCL stands for Marmara Credit Loops in English.

MCL or Marmara Coin is the first native Turkish coin based on a use case in real economy. The good thing is that the system can be used universally anywhere in the world as an international payment system for import and export.

As you know, cryptocurrencies are not widely yet used in the real economy for the exchange of goods and services. MCL, on the other hand, was developed for use in the real economy. You will see its power when you use it.

MCL is a project inspired by the way Post-dated Checks and promissory notes work, and its origins date back to 2013. Based on the "Town Story" told during the Ufuk Hattı TV programmes.

It is an Open Source and International project. It has been developed using Komodo technologies, which is considered Blockchain of Linux, to develop software in the Blockchain World.

Starting from 2013, 4 workshops were held at the university with academicians and business people from Marmara University and other universities, and after the needs were defined, software development was carried out.

The test chain studies of the project, which was developed in Marmara University Engineering Faculty Open Innovation Laboratory, were carried out in 2019 and the chain started its life on January 17, 2020 under the name of Marmara Blockchain.

MCL is Turkey's first Open Source native cryptocurrency project to be mined/staked based on the Proof of Work and Proof of Stake. It was designed to work on real economy.

MCL, also known as Marmara Coin, is based on the idea that buyers and sellers earn block rewards during shopping.

MCL can be staked on a simple MiniPC: No energy consumption and no need for expensive computers, as in other mining Bitcoin etc systems. It can be produced with a second hand PC, the computer you use in your daily work, or even a mini PC you buy especially for this purpose.

You can convert the MCLs you produce with Komodo, Bitcoin and other currencies on the Decentralized Exchange AtomicDEX, and convert them into Lira for purchasing goods and services directly on the market, trading jewelers or elsewhere.

****
Source: https://marmara.io/blog/marmara-chain-1/what-is-mcl-what-are-the-features-10

3x Staking Power: Up to 3x staking power on locks in credit loops

Block Reward Per Minute: The block time is 1 minute and the reward distributed per block is 30 MCL

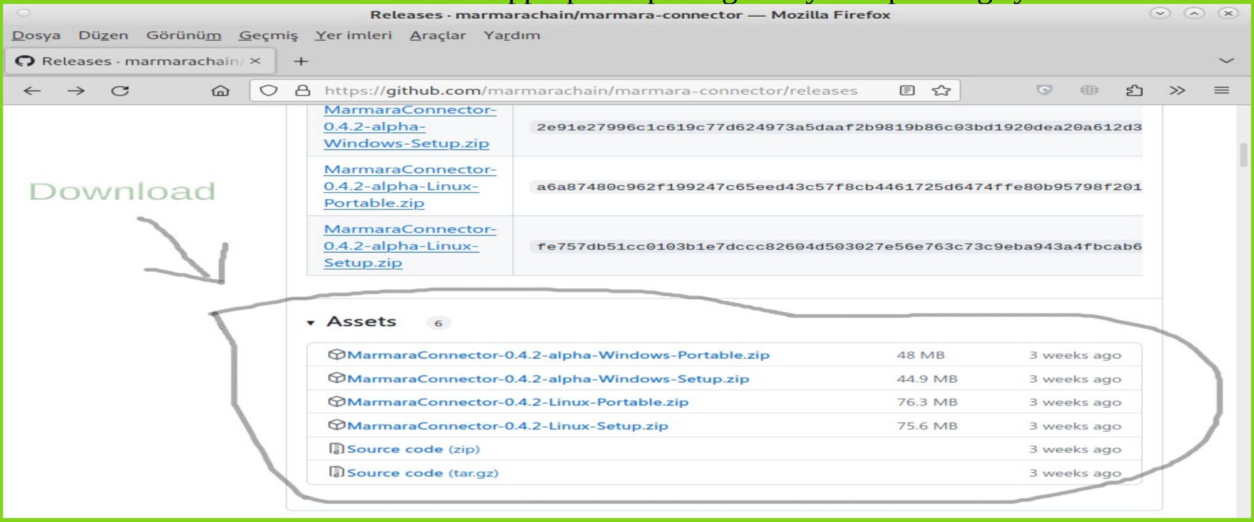
Pre-release: 2,000,000 MCL

Annual Supply : ~ 15,778,800 MCL

<u>INFORMATION NOTE</u>

The simplest way to install a Marmara Chain is to install using the latest version of Marmara Connector. Marmara Connector will check whether the Marmara Chain is installed in your system and if not, it will perform the installation process if you give approval. If you have installed the Marmara Chain on your operating system before installing the Marmara Connector, it will connect and enable you to easily perform your MCL operations through the graphical interface. If you encounter any problems during the installation process, you can get help and support from discord or telegram.

Download and install the appropriate package for your operating system


Releases · marmarachain/marmara-connector — Mozilla Firefox
Dosya Düzen Görünüm Geçmiş Yer imleri Araçlar Yardım
Releases · marmarachain/
https://github.com/marmarachain/marmara-connector/releases
Download

MarmaraConnector-0.4.2-alpha-Windows-Setup.zip	2e91e27996c1c619c77d624973a5daaf2b9819b86c03bd1920dea20a612d3
MarmaraConnector-0.4.2-alpha-Linux-Portable.zip	a6a87480c962f199247c65eed43c57f8cb4461725d6474ffe80b95798f201
MarmaraConnector-0.4.2-alpha-Linux-Setup.zip	fe757db51cc0103b1e7dccc82604d503027e56e763c73c9eba943a4fbcab6

Assets 6

MarmaraConnector-0.4.2-alpha-Windows-Portable.zip	48 MB	3 weeks ago
MarmaraConnector-0.4.2-alpha-Windows-Setup.zip	44.9 MB	3 weeks ago
MarmaraConnector-0.4.2-Linux-Portable.zip	76.3 MB	3 weeks ago
MarmaraConnector-0.4.2-Linux-Setup.zip	75.6 MB	3 weeks ago
Source code (zip)		3 weeks ago
Source code (tar.gz)		3 weeks ago

https://github.com/marmarachain/marmara-connector/releases

How Does MCL Work? How Can You Earn MCL?

HOW DOES MCL WORK?

The way MCL is working is explained in various articles. But let's explain it in a way that everyone can understand.

MCL working logic is very simple. The shortest and most natural way to earn MCL, that is, to produce MCL, is to use it in credit loops during shopping.

EARN WHILE SHOPPING

Let's say you are going to make a purchase as a buyer or seller by using another currency. When you pay in advance, the transaction closes there.

If you shop with MCL with credit loops, you earn as both a buyer and a seller. For example, you make and sell hobby products with resin. You tell the buyer that you can sell with MCL credit loop. If the customer pays in advance, he will not gain anything. You lock the payment to the Marmara blockchain for 100% as collateralization. You can earn MCL rewards with 3x staking power until the credit loop maturity date. Let's say you don't want to wait until the end of the credit's maturity. So you need cash. You will get the material for the resin. You can transfer the credit loop to the supplier (this is called endorsement in postdated checks). Now the person who sells the resin material along with your customer continues to earn with 3x staking rewards. You can also make new loops with another customer.

REINFORCING PURCHASING POWER OF CONSUMERS

MCL credit loops (MCL) is a system that supports the consumer, that is, the customer, and MCL reinforces the purchasing power of the customer. If the customer paid cash in advance, he would gain nothing. But when he did it with the credit loop, he got a chance to participate in 3x staking rewards. A useful system also for the seller; because you are now a manufacturer with high purchasing power customers. You can sell goods or services more easily.

When you decide to make a purchase in installments using the MCL credit loops method, the buyer (issuer) locks the money on the Marmara Blockchain with the terms and installments agreed between both the buyer and the seller, instead of making a cash payment. For each credit loop until the end of the maturity, both the buyer and the seller get a chance to find block rewards on the Marmara Blockchain with 3x staking power. You may have heard of Bitcoin mining. In fact, they earn a reward, namely Bitcoin, by performing a kind of notary public service. They need energy and expensive computing power to earn Bitcoin rewards. To earn the MCL Award, the buyer and seller need to make credit loops during shopping.

Therefore, MCL, Marmara Credit Loops, is a universal system as the first and only DeFi in the World designed to run in real economy to use during shopping. By using MCL Credit loops during shopping, both buyer, buyer and seller earn MCL Coin rewards.

****
Source: https://marmara.io/blog/marmara-chain-1/how-does-mcl-work-how-can-you-earn-mcl-9?anim=

Easy Installation of Marmara Blockchain and MCL Coin Production with Marmara Connector

Marmara Connector is an application developed for everyone to easily set up the Marmara Blockchain and start generating MCL coins.

WHAT IS MARMARA BLOCKCHAIN AND MCL?

Marmara Blockchain is a project inspired by credit instruments such as Forward Checks and Promissory Notes, which are produced within the shopping base. On the Marmara Blockchain, you can safely issue (discover) and circulate (turnover) all types of loans without problems such as dishonesty, forgery, indivisibility, and untraceability. Since MCL is a system developed for use in real economy, you can generate MCL while shopping.

Marmara Blockchain is a free blockchain and has its own coin called MCL. MCL on the Marmara Blockchain is not a token but a decentralized coin. The requirements are modest due to Marmara Energy and its processor-free 75% Proof-of-Stake algorithm (PoS). Therefore, Marmara Blockchain can be installed on an ordinary laptop, PC or even miniPC today and MCL coins can be generated on it.

Since MCL is a coin, it works as a kind of notary system. You can earn rewards by being a notary public on the Marmara blockchain. Approximately every minute, a block is issued and 30 MCLs as a block reward are distributed against notary public. You can earn block rewards by keeping your computer open by making credit cycles or locking it in active and keeping records. If you lock MCL in credit cycles to get the reward, you will have the chance to find the maximum reward with 3x staking. Credit Cycles can be used during shopping by endorsement (by transferring to another seller) within the shopping.

REQUIREMENTS TO INSTALL MARMARA BLOCKCHAIN

The requirements for setting up a Marmara Blockchain are very low. It is sufficient to have the following features:

- 64 Bit Operating System
- 8 GB of RAM
- Minimum 4 Cores CPU (Min. 6 cores recommended)
- SSD 100GB

WHAT IS MARMARA CONNECTOR?

Marmara Connector is an application developed for everyone to easily set up the Marmara Blockchain and start producing MCLs. Since it is necessary to install the blockchain to earn the MCL reward, the blockchain should be easily set up by anyone. That's why Marmara Connector was developed for this purpose.

Marmara Connector is very simple to install and use. It can be run on Windows and Linux Operating System locally on your machine or by installing it on a remote server.

MARMARA CONNECTOR INSTALLATION

Marmara Connector Installation is very simple.

Download and install the latest Marmara Connector Application from Marmara Chain Github Page. Installing the Marmara Connector itself is that simple.

See Releases · marmarachain/marmara-connector · GitHub

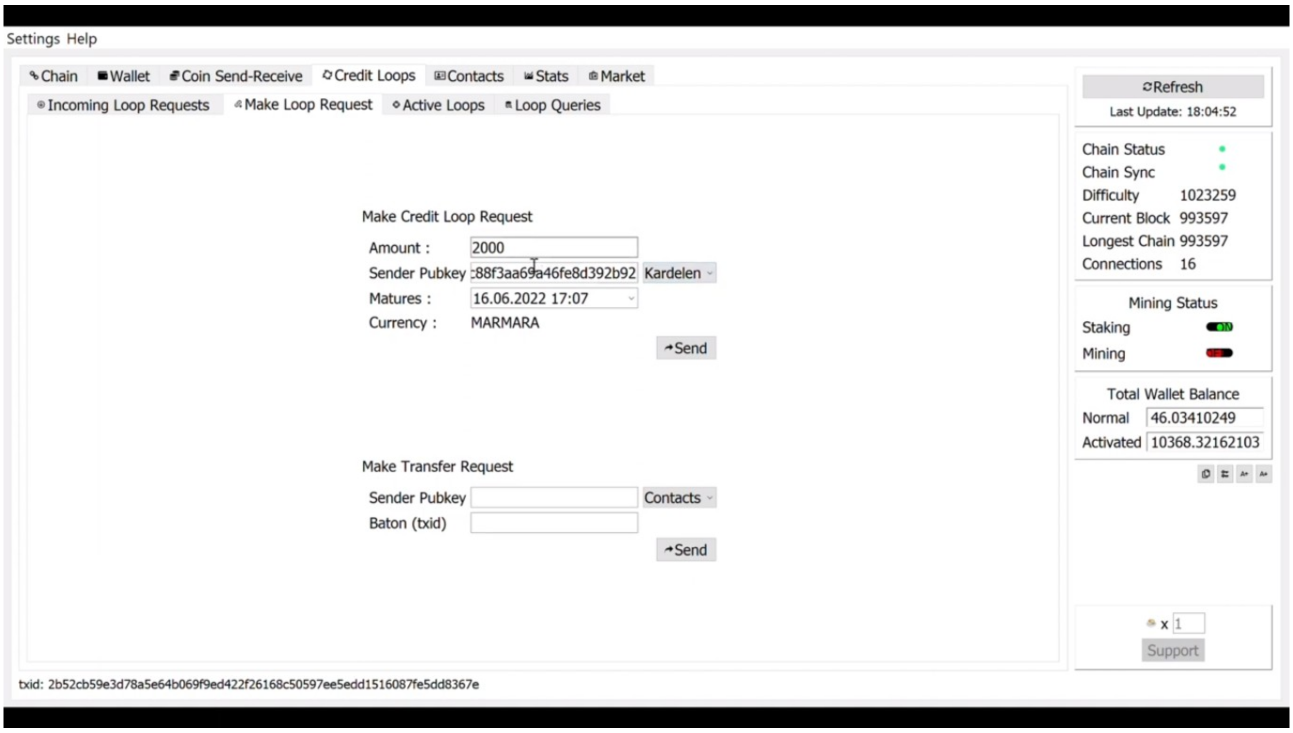
WHAT DOES MARMARA CONNECTOR DO?

Marmara Connector allows you to easily set up the Marmara Blockchain required for your notary public.

It allows you to make all kinds of transactions related to credit cycles while shopping.

It performs all other necessary transactions on Marmara Blockchain, transactions related to MCL coin wallet, and information functions.

The following are the operations that can be performed on the Connector when it is installed.



Last Update: 18:04:52

Make Credit Loop Request

Amount :	2000
Sender Pubkey	:88f3aa69a46fe8d392b92 Kardelen
Matures :	16.06.2022 17:07
Currency :	MARMARA

Make Transfer Request

Sender Pubkey	
Baton (txid)	

Chain Status	●
Chain Sync	●
Difficulty	1023259
Current Block	993597
Longest Chain	993597
Connections	16

Mining Status

Staking	ON
Mining	

Total Wallet Balance

Normal	46.03410249
Activated	10368.32162103

x 1

txid: 2b52cb59e3d78a5e64b069f9ed422f26168c50597ee5edd1516087fe5dd8367e

You can watch the video on the following link about Marmara Blockchain Installation Using Marmara Connector:
https://youtu.be/irb4y3iOsp8

BLOCKCHAIN SYNCHRONIZATION

Blockchain is a distributed notary public and unlike central database systems where the record is kept in only one place, each node (notary) keeps the same data. For this, he is entitled to a prize. Therefore, it is thousands of times safer where records are kept in a central location (eg banks etc.). You cannot perform notary transactions etc. on the chain without keeping all historical database records. Below are the synchronization procedures.

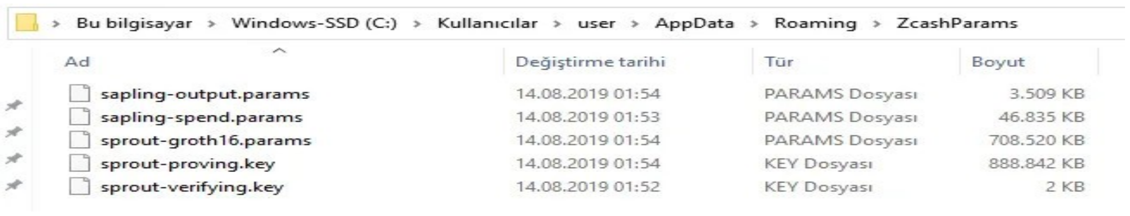
https://youtu.be/n0TG93mXjBk

POSSIBLE PROBLEMS DURING INSTALLATION OF MARMARA CONNECTOR

Marmara Connector does 3 things.

1. It downloads the necessary files to run the Marmara Blockchain. You must have the curl program installed to download it. If it is not installed on some Windows systems, it must be installed manually.

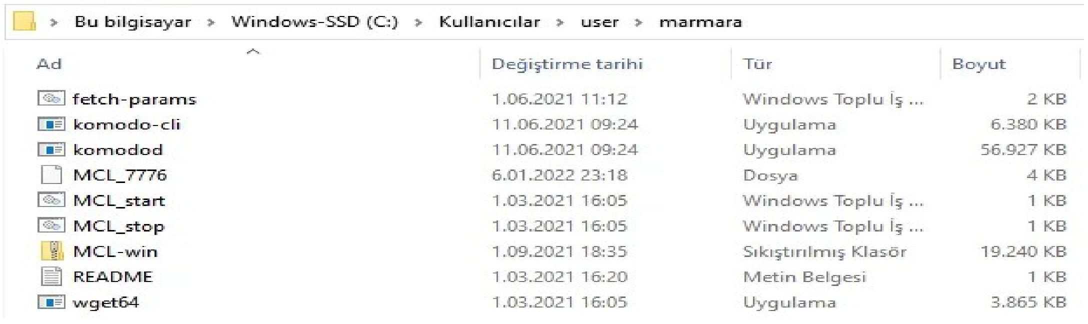
2.For Marmara Blockchain transactions, the following program files related to Marmara should be available under the C:\Users\user\AppData\Roaming\ZcashParams directory. Note: Pay attention to the files as below and their lengths.



> Bu bilgisayar > Windows-SSD (C:) > Kullanıcılar > user > AppData > Roaming > ZcashParams

Ad	Değiştirme tarihi	Tür	Boyut
sapling-output.params	14.08.2019 01:54	PARAMS Dosyası	3.509 KB
sapling-spend.params	14.08.2019 01:53	PARAMS Dosyası	46.835 KB
sprout-groth16.params	14.08.2019 01:54	PARAMS Dosyası	708.520 KB
sprout-proving.key	14.08.2019 01:54	KEY Dosyası	888.842 KB
sprout-verifying.key	14.08.2019 01:52	KEY Dosyası	2 KB

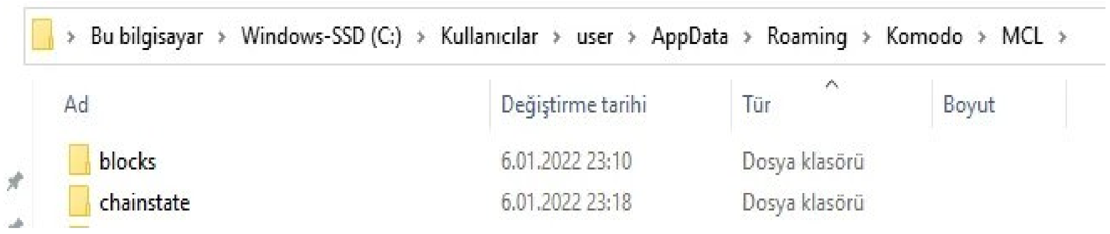
3. The programs required for all transactions related to Marmara Blockchain should be downloaded under the C:\Users\user\marmara folder as follows. Note: Pay attention to the files as below and their lengths. In the new version, there will be marmara-cli and marmarad files instead of komodo-cli and komodod files.



> Bu bilgisayar > Windows-SSD (C:) > Kullanıcılar > user > marmara

Ad	Değiştirme tarihi	Tür	Boyut
fetch-params	1.06.2021 11:12	Windows Toplu İş ...	2 KB
komodo-cli	11.06.2021 09:24	Uygulama	6.380 KB
komodod	11.06.2021 09:24	Uygulama	56.927 KB
MCL_7776	6.01.2022 23:18	Dosya	4 KB
MCL_start	1.03.2021 16:05	Windows Toplu İş ...	1 KB
MCL_stop	1.03.2021 16:05	Windows Toplu İş ...	1 KB
MCL-win	1.09.2021 18:35	Sıkıştırılmış Klasör	19.240 KB
README	1.03.2021 16:20	Metin Belgesi	1 KB
wget64	1.03.2021 16:05	Uygulama	3.865 KB

4. All data of historical blockchain records. The safest way to do this is to wait for it to be downloaded live. When you start the chain after installing it, it will be downloaded automatically in Marmara Connector. When all data is downloaded, it is said to be synchronous. You can sync quickly by downloading the daily updated Bootstrap file here for a quick download. Marmara Connector does this in itself. Bootstrap file should be opened under MCL folder in C:\Users\user\AppData\Roaming\Komodo\MCL directory. After stopping the synchronous chain, you can use the backup of the two folders when the chain data is corrupted.



> Bu bilgisayar > Windows-SSD (C:) > Kullanıcılar > user > AppData > Roaming > Komodo > MCL >

Ad	Değiştirme tarihi	Tür	Boyut
blocks	6.01.2022 23:10	Dosya klasörü	
chainstate	6.01.2022 23:18	Dosya klasörü	

Bootstrap backup is taken daily in https://explorer2.marmara.io/bootstrap

Alternative link :
https://eu.bootstrap.dexstats.info/MCL-bootstrap.tar.gz

Note for Bootstrap: Bootstrap only has the blocks and chainstate folder. Two is enough. After stopping the chain delete all existing folders except wallet.dat. Then open the two folders. You don't even need to download bootstrap. From time to time, stop the synchronized chain and back up the blocks and chainstate folders. Here are your own Bootstrap files.

You can get information from the following sources:
Marmara Website: https://marmara.io
Marmara Discord Group: https://marmara.io/discord
Twitter Marmara: https://twitter.com/marmarachain
Marmara Credit Loops Youtube Channel: https://www.youtube.com/channel/UC_Ym-tICCd7ATlBU_1ToEKw/videos

****
Sources:

- https://marmarachain-mcl.medium.com/marmara-connector-ile-marmara-blokzincirinin-kolay-kurulumu-ve-mcl-koin-%C3%BCretimi-1a1e56fadfa7
- https://marmara.io/discord

Build Marmara from Source

The basic Komodo software package includes two applications.

marmarad

The marmarad software application is the Smart Chain daemon that powers all Komodo blockchains.

marmara-cli

The marmara-cli software application allows a developer to execute API calls to marmarad via the command line.

Both are Installed Automatically

Both of these software applications are installed in the ~/komodo/src/ directory as a part of any of the following installation procedures.

Building Marmara Credit Loops Smart Chain From Source

One may also build Marmara Credit Loops Smart Chain from source. This is not required, however building from source is considered as the best practice in a production environment, since this allows one to instantly update to the latest patches and upgrades.

Linux

Requirements

- Linux (easiest with a Debian-based distribution, such as Ubuntu)
  For Ubuntu, we recommend using the Ubuntu 20.04, 22.04 or 24.04 LTS releases
- 64-bit Processor
- Minimum 4 Cores CPU (Min. 6 Cores CPU recommended)
- Minimum 8 GB of free RAM

Get Started

Verify that your system is up to date.

```
sudo apt-get update
sudo apt-get upgrade -y
```

Install the dependency packages

```
sudo apt-get install build-essential pkg-config libc6-dev m4 g++-multilib autoconf libtool ncurses-dev unzip git python python3-zmq zlib1g-dev wget curl bsdmainutils automake cmake clang ntp ntpdate nano -y
```

This action takes some time, depending on your Internet connection. Let the process run in the background. Once completed, follow the steps below to install Marmara Credit Loops smart chain.

Clone the Marmara Credit Loops Repository

```
cd ~
```

```
git clone https://github.com/marmarachain/marmara --branch master --single-branch
```

Skip the following steps and jump to Fetch the Zcash Parameters step if the system you are using has 8 GB or more RAM.

Changing swap size to 4 GB

Swap is a space on a disk which is used when the amount of physical RAM memory is full. If the system has 4 GB of RAM, to avoid cases of running out of RAM, swap space needs to be configured.

Check if OS already has swap enabled using sudo swapon --show . If swapfile exists and is not configured as at least 4 GB then turn off the swap before configuring it by sudo swapoff /swapfile

Execute the following commands to allocate 4 GB of RAM for the swap space.

```
sudo fallocate -l 4G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
```

Fetch the Zcash Parameters

```
cd marmara
./zcutil/fetch-params.sh
./zcutil/build.sh -j$(nproc)
```

Once this step is complete, you will find the marmarad & marmara-cli files in the src directory.

The following steps are optional to follow.

Setting up Swappiness

Swappiness is the kernel parameter that defines how much (and how often) Linux kernel copies RAM contents to swap. The following sets up the parameter's value as “10”. Note that the higher the value of the swappiness parameter, the more aggressively Linux's kernel will swap.

```
sudo sysctl vm.swappiness=10
```

This setting will persist until the next reboot. We can set this value automatically at restart by adding the line to our /etc/sysctl.conf file: sudo nano /etc/sysctl.conf vm.swappiness=10

Enabling UFW

This step is not required for installing MCL but can be used for server security purposes.

- Check the status of UFW by

```
sudo ufw status
```

- UFW package can be installed by executing the following command

```
sudo apt install ufw
```

- Activate UFW and allow connections by executing the following commands

```
echo y | sudo ufw enable
sudo ufw allow ssh
sudo ufw allow "OpenSSH"
sudo ufw allow 33824
sudo ufw allow out 33824
```

MacOS

Requirements

- OSX (version > 10.11)
- Minimum 4GB of free RAM (8GB+ recommended)

Ensure Command Line Tools are Installed

Issue the following command in a terminal.

```
xcode-select –install
```

Ensure brew is Installed

We use the software brew to install dependencies. If you have the latest version of brew installed already, you may skip this step.

```
usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew install/master/install)"
```

Use brew to Install Dependencies

Execute each command separately

```
brew update
brew upgrade
brew tap discoteq/discoteq; brew install flock
brew install autoconf autogen automake
brew install gcc@8
brew install binutils
brew install protobuf
brew install coreutils
brew install wget
```

Clone the Marmara Credit Loops Repository

```
cd ~
git clone https://github.com/marmarachain/marmara --branch master --single-branch
```

Fetch the Zcash Parameters

```
cd marmara
./zcutil/fetch-params.sh
./zcutil/build-mac.sh -j8
```

Once this step is complete, you will find the marmarad & marmara-cli files in the src directory.
Having completed these, launch the Marmara Chain by following the instructions in here.

Windows

The Windows software cannot be directly compiled on a Windows machine.

Rather, the software must be compiled on a Linux machine (Ubuntu recommended), and the respective output .exe must then be carried to the Windows machine.

Requirements

- Linux machine or Virtual Machine-based installation of Ubuntu Linux
- Linux Requirements in here.

Install the dependency packages

```
sudo apt-get install build-essential pkg-config libc6-dev m4 g++-multilib autoconf libtool ncurses-dev unzip git python python-zmq zlib1g-dev wget libcurl4-gnutls-dev bsdmainutils automake curl libsodium-dev cmake mingw-w64
```

Install Rust

```
curl https://sh.rustup.rs -sSf | sh
source $HOME/.cargo/env
rustup target add x86_64-pc-windows-gnu
```

Configure The Compiler to Use POSIX Thread Model

Execute the following command:

```
sudo update-alternatives --config x86_64-w64-mingw32-gcc
```

After having issued the above command, select the POSIX option as indicated below:

```
Selection   Path                                      Priority   Status
------------------------------------------------------------------------------------------------------
  0         /usr/bin/x86_64-w64-mingw32-gcc-win32     60         auto mode
  1         /usr/bin/x86_64-w64-mingw32-gcc-posix     30         manual mode
* 2         /usr/bin/x86_64-w64-mingw32-gcc-win32     60         manual mode

Press <enter> to keep the current choice[*], or type selection number: 1
```

Issue the following command:

```
sudo update-alternatives --config x86_64-w64-mingw32-g++
```

Having executing the above command, select the POSIX option as below:

```
There are 2 choices for the alternative x86_64-w64-mingw32-g++ (providing /usr/bin/x86_64-
w64-mingw32-g++)

  Selection    Path                                          Priority     Status
------------------------------------------------------------
    0          /usr/bin/x86_64-w64-mingw32-g++-win32     60           auto mode
    1          /usr/bin/x86_64-w64-mingw32-g++-posix     30           manual mode
* 2            /usr/bin/x86_64-w64-mingw32-g++-win32     60           manual mode

Press <enter> to keep the current choice[*], or type selection number: 1
```

Clone The Marmara Repository

```
git clone https://github.com/marmarachain/marmara --branch master --single-branch
cd marmara
```

Fetch the Zcash Parameters

```
./zcutil/fetch-params.sh
./zcutil/build-win.sh -j$(nproc)
```

Once this step is completed, you will find marmarad.exe & marmara-cli.exe files inside the src directory.

Copy and move these files to the Windows machine. Continue from here:
https://github.com/marmarachain/marmara/wiki/Install-Marmara-from-Precompiled-Binaries#windows

Getting Started with Marmara

How to get started with Marmara?

The basic Marmara software package includes two applications.

marmarad
The marmarad software application is the Smart Chain daemon that powers all Marmara blockchains.

marmara-cli
The marmara-cli software application allows a developer to execute API calls to marmarad via the command line.

MacOS

Find the directory where marmarad and marmara-cli are located. Open the command prompt and change to that directory.

Using marmara-cli and getinfo

To see general information about MCL Smart chain, execute the getinfo command below:

```
./marmara-cli -ac_name=MCL getinfo
```

When the returned properties of blocks and longestchain are equal to each other, the daemon is finished syncing with the network.

One can interact with the MARMARA blockchain in MacOS by issuing the MARMARA commands as in Linux OS.

Windows

Find the directory where marmarad.exe and marmara-cli.exe are located. Open the command prompt and change to that directory.

For instance, given that the respective files are located in Desktop under MCL folder; the command should be like below:

```
cd Desktop\MCL\
```

Using marmara-cli and getinfo

An example is given for getinfo command below:

```
marmara-cli -ac_name=MCL getinfo
```

Linux

If you have downloaded and build MCL from source then you can run the commands under komodo/src directory. According to your configuration, marmarad and marmara-cli may be under different directories. Hence, find where they are and change directory accordingly.

Launch the Marmara Chain with the following parameters:

```
./marmarad  &
```

Wait until it connects and synchronizes. You may check if the node synchronized to the chain by

executing the following:
```
./marmara-cli -ac_name=MCL getinfo
```

Indexing (Optional step for fastening up the process of downloading all the blocks)**

Newcomers need to wait for all the blocks to be downloaded to their machine. To fasten up this process, bootstrap may be downloaded and used.

Stop the Marmara blockchain by executing the following command:

Linux:
```
./marmara-cli -ac_name=MCL stop
```

Windows:
```
marmara-cli.exe -ac_name=MCL stop
```

To install bootstrap from the command line, execute the following command:

Linux:
```
wget https://eu.bootstrap.dexstats.info/MCL-bootstrap.tar.gz
```

Now, in the following command, tar will extract the bootstrap contents in a specific directory as shown below:
```
tar -xvf MCL-bootstrap.tar.gz -C .komodo/MCL
```

Now, relaunch the Marmara Chain by using the following command:

Linux:
```
./marmarad  &
```

Windows:
```
marmarad.exe  &
```

** Using Bootstrap is not a recommended approach to download the blocks since the bootstrap file may be distorted. The recommended approach is to index the blockchain from the start.**

Creating A Pubkey and Launching MCL with pubkey

To use Marmara Credit Loops, a user must have a pubkey and launch the chain with a pubkey. Otherwise, any mining or staking in the smart chain would be in vain. Since all mined or staked coins will also be sent to this address.

In order to get a pubkey, launch the Marmara Chain with the normal launch parameters and execute the getnewaddress API command:

Linux:

```
./marmara-cli -ac_name=MCL getnewaddress
```

Windows:

```
marmara-cli.exe -ac_name=MCL getnewaddress
```

getnewaddress command will in turn return a new address:

```
DO_NOT_USE_THIS_ADDRESSgg5jonaes1J5L786
```

Now, execute the validateaddress command.

Linux:

```
./marmara-cli -ac_name=MCL validateaddress
DO_NOT_USE_THIS_ADDRESSgg5jonaes1J5L786
```

Windows:

```
marmara-cli.exe -ac_name=MCL validateaddress
DO_NOT_USE_THIS_ADDRESSgg5jonaes1J5L786
```

This will return a json object with many properties. In the properties one can see:

```
"pubkey": "DO_NOT_USE_THIS_ADDRESS019n79b0921a1be6d3ca"
```

This will be your MCL pubkey, make sure to note it. You must now indicate it to the daemon. In order do this, first stop the daemon using the following command:

Linux:

```
./marmara-cli -ac_name=MCL stop
```

Windows:

```
marmara-cli.exe -ac_name=MCL stop
```

Launching MCL with pubkey

Relaunch your daemon using the required parameters, and make sure to include your pubkey as an additional parameter. For example:

Linux:

```
./marmarad -pubkey=DO_NOT_USE_THIS_ADDRESS019n79b0921a1be6d3ca &
```

Windows:

```
marmarad.exe  -pubkey=DO_NOT_USE_THIS_ADDRESS019n79b0921a1be6d3ca &
```

-genproclimit sets the number of threads to be used for mining.

If the parameter -genproclimit is set to 1 such as -gen -genproclimit=1 launches the Marmara Chain with single (1) CPU. Note that you can set this to any number of cores such as 3.

If the parameter -genproclimit is set to -1 such as -gen -genproclimit=-1

launches the Marmara Chain with the whole CPU.

If the parameter -genproclimit is set to 0 such as -gen -genproclimit=0 launches the Marmara Chain with Staking mode on and stakes on the Activated coin.

dumpprivkey

The dumpprivkey method reveals the private key corresponding to the indicated address. The command for this is given below for demo purposes.

Linux:
```
./marmara-cli -ac_name=MCL dumpprivkey "PWqwYaWNEVT7V9SdfFHARWnoB7vcpSfdvs"
```

Windows:
```
marmara-cli.exe -ac_name=MCL dumpprivkey
"PWqwYaWNEVT7V9SdfFHARWnoB7vcpSfdvs"
```

The response of the command gives the private address as in the example below:
```
DONOTUSETHISxxxxxxxxxxxxxxxxx7KkCmRnnSg7iXvRUqhYoxC9Y
```

importprivkey

The importprivkey method adds a private key (as returned by dumpprivkey) to your wallet. The command may take several arguments as described in here. The simplest form of command for this is given below for demo purposes.

Linux:
```
./marmara-cli -ac_name=MCL importprivkey
"DONOTUSETHISxxxxxxxxxxxxxxxxx7KkCmRnnSg7iXvRUqhYoxC9Y"
```

Windows:
```
marmara-cli.exe -ac_name=MCL importprivkey
"DONOTUSETHISxxxxxxxxxxxxxxxxx7KkCmRnnSg7iXvRUqhYoxC9Y"
```

The response of the command gives the wallet address as in the demo response below:
```
R9z796AehK5b6NCPeVkGUHSpJnawerf8oP
```

Now, the wallet address can be validated through validateaddress to get the MCL pubkey:

Linux:
```
./marmara-cli -ac_name=MCL validateaddress R9z796AehK5b6NCPeVkGUHSpJnawerf8oP
```
Windows:
```
marmara-cli.exe -ac_name=MCL validateaddress R9z796AehK5b6NCPeVkGUHSpJnawerf8oP
```

After getting the respective MCL pubkey from the response of the command above, relaunch the Marmara Chain with that pubkey as in here .

Creating a Wallet Address from One's own Words in Marmara Blockchain

For Linux OS; as previously explained, one can make use of the command `./marmara-cli -ac_name=MCL getnewaddress` in order to get a wallet address and then use the command `./marmara-cli -ac_name=MCL validateaddress "Rwalletno"` to get the respective Privkey.

However, the Privkey received in this way consists of a hard to remember combination of letters and number. Note that keeping this privkey safe and secure and reachable only by you is very important and essential way to reach your assets. Hence, one can generate a privkey from his/her own set of words and keep a record of these words to generate the respective privkey whenerever needed.

For this purpose, create a set of 12 seed words that have no special characters in them. For instance: "can create your own keywords and generate privkey from these whenever needed".

Then use the following command to get the respective privkey generated:

Linux:

```
./marmara-cli -ac_name=MCL convertpassphrase "can create your own keywords and generate privkey from these whenever needed"
```

Windows:

```
marmara-cli.exe -ac_name=MCL convertpassphrase "can create your own keywords and generate privkey from these whenever needed"
```

The command in turn returns the following JSON Object:

```
{
  "agamapassphrase": "can create your own keywords and generate privkey from these whenever needed",
  "address": "RCna416S62uNVAKMwRuFtmRH2X3xJuTidD",
  "pubkey": "03a3c416d3bafe44448dcb3a56e5dd13db5cc3e6992051a61a2f5675b4796c0d7a",
  "privkey": "9005f5b2593ab3e662e1330ceca848d62fb2021165c6484ac7037cb4efc5317e",
  "wif": "UtqWJyhXjJWYigKyKCm5ML96LvSZFDfZrT2HYXAdGs8byZ3uCMzy"
}
```

Later, one can use importprivkey method to add the respective private key to the wallet address:

Linux:

```
./marmara-cli -ac_name=MCL importprivkey "UtqWJyhXjJWYigKyKCm5ML96LvSZFDfZrT2HYXAdGs8byZ3uCMzy"
```

Windows:

```
marmara-cli.exe -ac_name=MCL importprivkey "UtqWJyhXjJWYigKyKCm5ML96LvSZFDfZrT2HYXAdGs8byZ3uCMzy"
```

Now, the owner of the assets needs to keep a record of the keyword combination safely to generate the respective privkey whenever needed. Remember that private keys should always be kept secret and so are the keywords!

Taking Backup of Wallet

Backing up the wallet.dat file is very essential as it holds the assets of one.

Linux

On a Linux machine, the file could be found in: ~/.komodo/MCL/wallet.dat

One method to backup this file is to archive a copy of the file.

```
#Copy the wallet.dat file
cp -av ~/.komodo/MCL/wallet.dat ~/wallet.dat

#Rename the wallet.dat file
mv ~/wallet.dat ~/2020-08-09-wallet_backup.dat

# Make an archieve of the wallet.dat file
tar -czvf ~/2020-08-09-wallet_backup.dat.tgz ~/2020-08-09-wallet_backup.dat

# Move the final file to a secure location
```

Checking the Staking/Mining mode of your node in Marmara Chain

The following command helps to check the mode of the node:

Linux:

```
./marmara-cli -ac_name=MCL getgenerate
```

Windows:

```
marmara-cli.exe -ac_name=MCL getgenerate
```

Once the command given above is executed, the following JSON object gets returned:

```
{
  "staking": false,
  "generate": true,
  "numthreads": 1
}
```

From above, it can be seen that the node is in the mining node.

"staking": false means that staking is off.

"generate": true means that mining is active.

"numthreads": 1 refers to the number of cores used for mining. In this case, this parameter was set to one earlier.

To change mode of the node to staking, execute the command below:

Linux:

```
./marmara-cli -ac_name=MCL setgenerate true 0
```

Windows:

```
marmara-cli.exe -ac_name=MCL setgenerate true 0
```

Now checking the status of the node:

Linux:

```
./marmara-cli -ac_name=MCL getgenerate
```

Windows:

```
marmara-cli.exe -ac_name=MCL getgenerate
```

This command returns a JSON object:

```
{
  "staking": true,
  "generate": false,
  "numthreads": 0
}
```

Note that "staking": true will not be of use if you have no activated coins!

Sending Coins to an Address

One may directly send a payment to a given address by issuing the following command. The payment is made based on the Normal Amount available on the respective wallet. The amount is rounded to the nearest 0.00000001. A transaction fee is deducted for the transaction being made from one's Normal Amount.

Linux:

```
./marmara-cli -ac_name=MCL sendtoaddress "MCL_address" amount
```

Windows:

```
marmara-cli.exe -ac_name=MCL sendtoaddress "MCL_address" amount
```

Note: This command directly sends payment to the specified address and should be used carefully. As, the payment sent through this method cannot be redeemed later.

Reaching Details about Node and Wallet in Marmara Chain

The following commands are useful for getting details about your node and wallet.

- getinfo

Linux:

```
./marmara-cli -ac_name=MCL getinfo
```

Windows:

```
marmara-cli.exe -ac_name=MCL getinfo
```

A typical result is given below. Note that the **synced status** is false:

```
{
  "version": 3000300,
  "protocolversion": 170009,
  "KMDversion": "0.5.3",
  "synced": false,
  "notarized": 0,
  "prevMoMheight": 0,
  "notarizedhash":
"0000000000000000000000000000000000000000000000000000000000000000",
  "notarizedtxid":
"0000000000000000000000000000000000000000000000000000000000000000",
  "notarizedtxid_height": "mempool",
  "KMDnotarized_height": 0,
  "notarized_confirms": 0,
  "walletversion": 60000,
  "balance": 0.00000000,
  "blocks": 72200,
  "longestchain": 586375,
  "tiptime": 1583613252,
  "difficulty": 38227.25456170298,
  "keypoololdest": 1614160564,
  "keypoolsize": 101,
  "paytxfee": 0.00000000,
  "sapling": 61,
  "timeoffset": 0,
  "connections": 16,
  "proxy": "",
  "testnet": false,
  "relayfee": 0.00000100,
  "errors": "",
  "CCid": 2,
  "name": "MCL",
  "p2pport": 33824,
  "rpcport": 33825,
  "magic": -1687041972,
  "premine": 2000000,
  "reward": "3000000000",
  "halving": "0",
  "decay": "0",
  "endsubsidy": "0",
  "notarypay": "0",
  "staked": 75
}
```

getinfo command returns important details such as the version of MARMARA through "version"; synchronization status of your node through synced (this parameter's value is true if the parameters "blocks" and "longestchain" are equal); difficulty of the chain through "difficulty"; number of nearest connected nodes to the chain through "connections"; the pubkey with which you are connected to the chain through "pubkey":

- getpeerinfo

getpeerinfo command returns detailed information on nearest connected nodes to the chain around your node.

Linux:
```
./marmara-cli -ac_name=MCL getpeerinfo
```

Windows:
```
marmara-cli.exe -ac_name=MCL getpeerinfo
```

- marmarainfo

Linux:
```
./marmara-cli -ac_name=MCL marmarainfo 0 0 0 0 pubkey
```

Windows:
```
marmara-cli.exe -ac_name=MCL marmarainfo 0 0 0 0 pubkey
```

marmarainfo command returns important details such as the normal amount in the pubkey through "myPubkeyNormalAmount"; the activated amount through "myActivatedAmount"; the details of credit loops made through "Loops"; the total amount locked in credit loops through "TotalLockedInLoop"; the number of credit loops closed through "numclosed"; and the details of credit loops closed through "closed".

Activating and Deactivating Coins

- marmaralock

marmaralock is used to activate the coins. Active coins are needed for staking and if there are none then even if the staking mode is on, no blocks would be found through staking. The entire command is given below and the amount is to be replaced by the amount of coins such as 1000.

Linux:
```
./marmara-cli -ac_name=MCL marmaralock amount
```

Windows:
```
marmara-cli.exe -ac_name=MCL marmaralock amount
```

This command in turn returns a JSON object with the result of the operation and the hex created for it. Note that the "hex" given below is for demonstration purposes.

```
{
  "result": "success",
  "hex":
"0400008085202f89020039b219200ae4b5c83d77bffce7a8af054d6fb..........e9181f6aac3e1beb1e26
0e9a1f49ed24e6ac00000000edeb04000000000000000000000000"
}
```

Now, in order to confirm this transaction, copy the hex returned through the JSON object and validate it through the sendrawtrasaction command given below:

Linux:

```
./marmara-cli -ac_name=MCL sendrawtransaction
0400008085202f89020039b219200ae4b5c83d77bffce7a8af054d6fb..........e9181f6aac3e1beb1e26
0e9a1f49ed24e6ac00000000edeb04000000000000000000000000
```

Windows:

```
marmara-cli.exe -ac_name=MCL sendrawtransaction
0400008085202f89020039b219200ae4b5c83d77bffce7a8af054d6fb..........e9181f6aac3e1beb1e26
0e9a1f49ed24e6ac00000000edeb04000000000000000000000000
```

If the above command gets successfully executed in the blockchain, it gives out a transaction id in response. One may check if this transaction is verified by searching the respective id in the Marmara Explorer site. To see the activated coins, use marmarainfo command provided earlier and search for the value across the "myActivatedAmount" parameter. Note that the raw transactions are collected in the mempool and a few blocks may be needed to found to see the transaction recorded on the block.

marmaraunlock is used deactivate the coins i.e. turn them into normal amount. Normal Amount is utilized for sending payments directly to an address through sendtoaddress command explained earlier. The amount is to be replaced by the amount of coins to be deactivated such as 500.

Linux:

```
./marmara-cli -ac_name=MCL marmaraunlock amount
```

Windows:

```
marmara-cli.exe -ac_name=MCL marmaraunlock amount
```

In the same way explained earlier, this transaction needs to be validated through the sendrawtrasaction command given above. For this purpose, copy the hex returned by marmaraunlock command and use it with sendrawtrasaction command.

- listaddressgroupings

listaddressgroupings is used to list the pairs of wallet addresses and respective normal amounts in them. The usage is given in the command below.

Linux:

```
./marmara-cli -ac_name=MCL listaddressgroupings
```

Windows:

```
marmara-cli.exe -ac_name=MCL listaddressgroupings
```

- Marmaraamountstat

This command is used to get all the activated, normal and locked in loop amount of the Marmara blockchain irrespective of the owner. If the begin_height and end_height inputs are set to zero(0), it gets the entire set of data from blocks.

Linux:

```
./marmara-cli -ac_name=MCL marmaraamountstat begin_height end_height
```

Windows:

```
marmara-cli.exe -ac_name=MCL marmaraamountstat begin_height end_height
```

Example for Linux OS:

```
./marmara-cli -ac_name=MCL marmaraamountstat 50000 60000
```

Sample Result:

```
{
   "result": "success",
   "BeginHeight": 50000,
   "EndHeight": 60000,
   "TotalNormals": 166854.85464948,
   "TotalPayToScriptHash": 0.00000000,
   "TotalActivated": 582681.11953150,
   "TotalLockedInLoops": 237510.00000000,
   "TotalUnknownCC": 0.02820000,
   "SpentNormals": 161377.88109325,
   "SpentPayToScriptHash": 0.00000000,
   "SpentActivated": 382128.12128773,
   "SpentLockedInLoops": 143510.00000000,
   "SpentUnknownCC": 0.00000000
}
```

help

This command is used to get all the methods related to the Marmara smart chain.

Linux:
```
./marmara-cli -ac_name=MCL help
```

Windows:
```
marmara-cli.exe -ac_name=MCL help
```

One can also check out a specific method's parameters by issuing the following command with the method name mentioned: For example:

Linux:
```
./marmara-cli -ac_name=MCL help getinfo
```

Windows:
```
marmara-cli.exe -ac_name=MCL marmaraamountstat
```

****

Source: https://github.com/marmarachain/marmara/wiki/Getting-Started-with-Marmara

References

For more detailed information on Komodo Antara Framework and its details, please refer to its extended developer documentation.
https://developers.komodoplatform.com/

How to make Marmara Credit Loops

The current Marmara Credit loops currently work based on Protocol 1 which is in 100% collateralization mode. 100 % collateralization is made by issuer on behalf of both himself/herself and holder. Both issuer and holder have the 3x staking chance to get blockchain rewards. Issuer has the 3x staking chance until maturity date of credit whereas holder has the 3x staking chance until he/she endorses/transfers the credit to a new holder who will continue staking with the issuer. The Credit loops can be made using only activated coins.

Terminology

Issuer: The person who first creates a credit in a credit loop. It is the person who forms the first node in the credit loop. The credit may be collateralized 100% or with even zero-collateralization until maturity date of a credit.

Bearer (Holder): The last node in a credit loop is always called bearer (holder). When someone transfers a credit in a loop, that node becomes immediately an endorser.

Endorser: All other nodes that fall between the issuer, which is the first node in the credit loop, and the last node, and transfer the credit to the next node.

Maturity: The time a credit expires. It is measured as blocks in estimation. The block per day is 1440 (60 blocks an hour times 24 hours a day). Suppose a credit is for 100 days, the maturity is 1440x100, i.e. 144,000 blocks.

Settlement: When a credit matures, settlement is made to the holder, the last node in a loop. Settlement may be automatic or manual.

Escrow: Trust based parties for Protocol 2. If EscrowOn is false, then 100% collateralization is used and settlement is automatic. There is no need for escrows in protocol 1 which works as complete trustless version.

Avalist: Avalists support issuer or endorsers with MCL as additional colletarization and can earn with 3x staking with those support. In case of non-redemption, their funds are utilized. Avalists are available only in Protocol 2. The parameter avalcount is always zero for protocol 1.

BlockageAmount: This is a term for protocol 2. An issuer may be asked to put some collateralization by a holder. In that case, the issuer gets benefit of 3x staking in protocol 2.

Dispute Expiry: It is grace period for solving non-redemption problem in credit loops in protocol 2. An issuer may have this time as blocks when creating a credit under protocol 2 without or insufficient collateralization. Before this period expires, an escrow should do all actions according to aggrement with the issuer to solve non-redemption. Otherwise, the escrow is penalized in the system.

Important Commands for Making Credit Loops

- *marmarareceive*

This command is used to get a credit from an issuer or an endorser. When asking a credit from an issuer, i.e. the first node, it has a unique use. In other nodes, it is the same.

Scenario 1: Two nodes are making a credit loop for the first time. This credit loop may be created for a sale of a good or service in the market. In such case, the holder (the one selling the product/service) should request for a credit from the issuer (the one paying the product/service) by writing down the following command:

Linux:

```
./marmara-cli -ac_name=MCL marmarareceive senderpk amount currency matures
'{"avalcount":"n"}'
```

Windows:

```
marmara-cli.exe -ac_name=MCL marmarareceive senderpk amount currency matures
{\"avalcount\":\"n\"}
```

senderpk is the pubkey address of the issuer (the one paying the product/service)

amount is the payment amount. Please note that this amount should be available in activated fund of the issuer and if not then must be activated thru marmaralock command by th issuer.

currency is MARMARA

matures is the time that respective credit expires, 60 blocks an hour times 24 hours a day making 1440 blocks per day.

'{"avalcount":"n"}' is the number of avalists i.e. '{"avalcount":"0"}' for protocol 1. Replace n with 0 for Protocol 1.

This marmarareceive call generates a hex code. This HEXCODE needs to be verified by the holder by executing the sendrawtransaction command:

Linux:

```
./marmara-cli -ac_name=MCL sendrawtransaction HEXCODE
```

Windows:

```
marmara-cli.exe -ac_name=MCL sendrawtransaction HEXCODE
```

Once this command is executed, a transaction id named txid gets generated. This txid along with the receiverpk needs to be communicated to the issuer to complete the credit loop. But, an alternative to this communication would be the use of marmarareceivelist method which could be used to see the receive requests made to the issuer himself/herself.

marmarareceivelist method would be executed by the issuer by the following command:

Linux:

```
./marmara-cli -ac_name=MCL marmarareceivelist pubkey maxage
```

Windows:

```
marmara-cli.exe -ac_name=MCL marmarareceivelist pubkey maxage
```

pubkey is the pubkey address of the issuer connected to the Marmara Chain and the maxage by default is 24x60 The response of this command is a list of pair of txid's created by the respective pubkeys for a given max age.

- *marmaraissue*

This command is only used by issuer, the first node to create/issue a credit. By this, a credit is also transferred to the first holder, i.e. the second node. Many of the parameters for a credit loop is decided between the issuer and the first holder. marmaraissue method takes in the following arguments:

Linux:

```
./marmara-cli -ac_name=MCL marmaraissue receiverpk '{"avalcount":"n",
"autosettlement":"true"|"false", "autoinsurance":"true"|"false", "disputeexpires":"offset",
"EscrowOn":"true"|"false", "BlockageAmount":"amount" }' requesttxid
```

Windows:

```
marmara-cli.exe -ac_name=MCL marmaraissue receiverpk
"{\"avalcount\":\"n\", \"autosettlement\":\"true"|"false\", \"autoinsurance\":\"true"|"false\", \"disput
eexpires\":\"offset\", \"EscrowOn\":\"true"|"false\", \"BlockageAmount\":\"amount\"}" requesttxid
```

receiverpk is the pubkey of the receiver which is the holder.

"avalcount":"n" is the number of avalists i.e. '{"avalcount":"0"}' for protocol 1.

"autosettlement":"true"|"false" AutoSettlement is true due to 100% collateralization in Protocol 1.

"autoinsurance":"true"|"false" Autoinsurance is true due to 100% collateralization in Protocol 1.

"disputeexpires":"offset" Dispute expiry is set to 0 due to 100 collateralization in Protocol 1.

"EscrowOn":"true"|"false" EscrowOn is set to false due to 100% collateralization in Protocol 1.

"BlockageAmount":"amount" } blockage amount is set to 0 due to 100 collateralization in Protocol 1.

requesttxid is the txid generated by the holder communicated to the issuer.

A typical example of a marmaraissue command to complete the credit loop by the issuer is given below:

Linux:

```
./marmara-cli -ac_name=MCL marmaraissue receiverpk '{"avalcount":"0",
"autosettlement":"true", "autoinsurance":"true", "disputeexpires":"0", "EscrowOn":"false",
"BlockageAmount":"0" }' requesttxid
```

Windows:

```
marmara-cli.exe -ac_name=MCL marmaraissue receiverpk
"{\"avalcount\":\"0\" \"autosettlement\":\"true\" \"autoinsurance\":\"true\" \"disputeexpires\":\"offse
t\" \"EscrowOn\":\"false\" \"BlockageAmount\":\"0\"}" requesttxid
```

This marmaraissue command in turn returns a hex code response, and now the issuer has to execute the sendrawtransaction method to get the transaction executed on the blockchain as follows:
Linux:

```
./marmara-cli -ac_name=MCL sendrawtransaction HEXCODE
```

Windows:

```
marmara-cli.exe -ac_name=MCL sendrawtransaction HEXCODE
```

This creates a credit loop between the issuer and the holder. The credits locked in a loop can be circulated to buy things during shopping. The issuer and the holder get 3 times of chances of staking on the MCL funds until the maturity of the credit loop.

- *marmaracreditloop*

To display the credit loop between the issuer and the holder, the following marmaracreditloop command may be executed:
Linux:

```
./marmara-cli -ac_name=MCL marmaracreditloop txid
```

Windows:

```
marmara-cli.exe -ac_name=MCL marmaracreditloop txid
```

txid is the baton transfer id of the Marmara Credit Loop.

- *marmaratransfer*

Scenario 2: The holder from the previous scenario wishes to utilize the coins locked in loop by buying goods/services on the same credit loop created earlier. For such case, when the holder transfers a credit in a loop, that node immediately becomes an endorser. And in such way, the last node in a credit loop is always called the bearer (holder). In other words, all endorsers are previously holders. One should bear in mind that endorsers lose the 3x staking power when a credit is transferred to a new holder.

For this purpose, the new holder makes a marmarareceive request to the endorser to get the credit for selling the goods/services by the following command:
Linux:

```
./marmara-cli -ac_name=MCL marmarareceive senderpk batontxid '{"avalcount":"n"}'
```

Windows:

```
marmara-cli.exe -ac_name=MCL marmarareceive senderpk batontxid {\"avalcount\":\"n\"}
```

senderpk is the pubkey address of the endorser (the one buying the product/service)

batontxid is the baton transaction id of the previously created credit loop

{"avalcount":"n"} is the number of avalists i.e. {"avalcount":"0"} for protocol 1.

This marmarareceive call generates a hex code. This HEXCODE needs to be verified by the new holder by executing the sendrawtransaction command:

Linux:

```
./marmara-cli -ac_name=MCL sendrawtransaction HEXCODE
```

Windows:

```
marmara-cli.exe -ac_name=MCL sendrawtransaction HEXCODE
```

Once this command is executed, a transaction id named txid gets generated. This txid along with the receiverpk needs to be communicated to the endorser to complete the credit loop. But, an alternative to this communication would be the use of marmarareceivelist method which could be used by the endorser to see the receive requests made to himself/herself. Then, the endorser executes the following marmaratransfer command to get the credits transferred to the new holder:

Linux:

```
./marmara-cli -ac_name=MCL marmaratransfer receiverpk '{"avalcount":"n"}' requesttxid
```

Windows:

```
marmara-cli.exe -ac_name=MCL marmaratransfer receiverpk {\"avalcount\":\"n\"} requesttxid
```

receiverpk is the pubkey of the receiver which is the new holder.

"avalcount":"n" is the number of avalists i.e. '{"avalcount":"0"}' for protocol 1 (n=0).

requesttxid is the txid generated by the new holder communicated to the endorser.

Then the endorser executes the sendrawtransaction command with the hex code resulting from marmaratransfercommand.

Please note that sendrawtransaction command is used after marmarareceive, marmaraissue, marmaratransfer and marmaralock to make the results of commands to be executed on blockchain. The usage is presented throughout the scenario 1 and 2.

In this way, the credit loops can circulate up to 1000th node within the maturity time to buy goods/services.

- *marmaraholderloops*

This command returns the open and closed loops which can be constrainted by setting the first and last height and the minimum and maximum amount. The method returns all the available data if all the parameters are set to zero. marmaraholderloops command is indicated below:

Linux:

```
./marmara-cli -ac_name=MCL marmaraholderloops firstheight lastheight minamount maxamount pubkey
```

Windows:

```
marmara-cli.exe -ac_name=MCL marmaraholderloops firstheight lastheight minamount maxamount pubkey
```

****
Source: https://github.com/marmarachain/marmara/wiki/How-to-make-Marmara-Credit-Loops

References

For more detailed information on how Marmara Credit Loops work, kindly refer to detailed article here: https://medium.com/@drcetiner/how-marmara-credit-loops-mcl-work-31d1896190a5

Installing Marmara from Precompiled Binaries

One can download and unzip our pre-compiled binaries. This is the simplest method and hence requires no compiling procedure.

Linux

Verify that your system is up to date.

```
sudo apt-get update
sudo apt-get upgrade -y
```

Install the dependency packages

```
sudo apt-get install libgomp1
```

Downloading Assets from Releases

Go to Releases and download Marmara linux binaries .zip from Assets and follow the instructions given below.

Make sure to follow the following instructions on the downloaded directory or any other directory where you moved or copied.

```
sudo apt install unzip
unzip MCL-linux.zip
sudo chmod +x marmarad marmara-cli fetch-params.sh
./fetch-params.sh
```

Start and Run the Marmara Chain

To start the chain, open the command prompt and execute the following sets of commands:

```
./marmarad
```

To stop the chain:

```
./marmara-cli -ac_name=MCL stop
```

Backup Your Wallet

Backing up the wallet.dat file is very essential as it holds the assets of one. On a Linux machine, the file could be found in: ~/.komodo/MCL/wallet.dat

One method to backup this file is to archive a copy of the file.

```
#Copy the wallet.dat file
cp -av ~/.komodo/MCL/wallet.dat ~/wallet.dat

#Rename the wallet.dat file
mv ~/wallet.dat ~/2020-08-09-wallet_backup.dat

# Make an archieve of the wallet.dat file
tar -czvf ~/2020-08-09-wallet_backup.dat.tgz ~/2020-08-09-wallet_backup.dat

# Move the final file to a secure location
```

Enabling UFW

This step is not required for installing MCL but can be used for server security purposes.

- Check the status of UFW by

```
sudo ufw status
```

- UFW package can be installed by executing the following command

```
sudo apt install ufw
```

- Activate UFW and allow connections by executing the following commands

```
echo y | sudo ufw enable
sudo ufw allow ssh
sudo ufw allow "OpenSSH"
sudo ufw allow 33824
sudo ufw allow out 33824
```

MacOS

Issue the following command in a terminal to install the command line tools:

```
xcode-select –-install
```

Download the .zip file from Releases page under Assets here to your macOS machine and extract the files. Change directory to the location of marmarad & marmara-cli on the terminal.

Run the following command:

```
./fetch-params.sh
```

Start and Run the Marmara Chain

To start the chain, open the command prompt and execute the following sets of commands:

```
./marmarad
```

To stop the chain:

```
./marmara-cli -ac_name=MCL stop
```

Backup Your Wallet

Backing up your wallet.dat file is very critical. One method to backup this file is to archive a copy of the file.

On MacOS, the wallet.dat file is located here: ~/Library/Application\ Support/Komodo/wallet.dat

Windows

Downloading Files from Releases

Download the .zip file from Releases page under Assets here to your Windows computer and extract&place the files (marmarad.exe & marmara-cli.exe) in a new folder on the Desktop called MCL or any other location you prefer (remember the location and use that).
For this guide, we are using MCL directory on Desktop.

Create the Directory for the Zcash Parameters

Open the command line prompt and execute the following command:

```
mkdir “%HOMEPATH%\AppData\Roaming\ZcashParams”
```

Download following files and move them into the newly created directory i.e. MCL.

- sprout-proving.key
- sprout-verifying.key
- sapling-spend.params
- sapling-output.params
- sprout-groth16.params

Start and Run the Marmara Chain

To start the chain, open the command prompt and execute the following sets of commands:

```
cd Desktop\MCL\

marmarad.exe -ac_name=MCL -ac_supply=2000000 -ac_cc=2 -addnode=5.189.149.242 -
addnode=161.97.146.150 -addnode=149.202.158.145 -addressindex=1 -spentindex=1 -
ac_marmara=1 -ac_staked=75 -ac_reward=3000000000
```

To stop the chain:

```
cd Desktop\MCL\
marmara-cli.exe -ac_name=MCL stop
```

Backup Wallet

Backing up the wallet.dat file is very critical. One method to backup this file is to create a copy and archive it.

On Windows machine, the file is located below:

%HOMEPATH%\AppData\Roaming\Komodo\MCL\wallet.dat

Making Marmara Credit Loops

Start the chain as with pubkey as instructed in the usage guideline here or https://github.com/marmarachain/marmara/wiki/Getting-Started-with-Marmara

****

Source: https://github.com/marmarachain/marmara/wiki/Install-Marmara-from-Precompiled-Binaries

Understanding Marmara Chain

Marmara Chain stands out in the blockchain world with its hybrid consensus mechanism, employing 25% Proof of Work (PoW) and 75% Proof of Stake (PoS). This innovative blend ensures secure, decentralized control while promoting an efficient and scalable network. Most blockchains use Proof of Work (PoW) or Proof of Stake (PoS) consensus mechanisms. PoW involves solving complex puzzles to add a new block to the blockchain (think of it like a digital mining process). PoS, on the other hand, involves holding and locking up a certain amount of tokens to validate a new block (imagine notaries getting rewards based on their shares in a common fund but with a self-financing mechanism).

The Marmara Credit Loops (MCL) Solution

At the heart of Marmara Chain is MCL, a system that brings unprecedented liquidity and utility to traditional financial instruments even before their maturity dates. But that's not all. MCL takes the unique approach of 'activating' and 'locking' coins for staking, providing varying degrees of staking power - 1x for activated and a whopping 3x for those locked within credit loops. The best part? Activated coins can be unlocked anytime, offering a level of liquidity unheard of in traditional staking systems while coins locked in credit loops can still be circulated in real economy by simply endorsing same as post-dated checks.

Revolutionizing Transactions with Credit Loops

Credit loops act as digital versions of post-dated checks or promissory notes, offering unique benefits. Coins locked in credit loops not only boast 3x staking power but can also be circulated in the real economy simply by endorsing them. This way, they continue contributing to the staking power of the holder while serving real-world transactions. Upon the maturity date, they are automatically unlocked.

Mining and Earning MCL Coins

MCL coins, the native currency of Marmara Chain, can be earned in more ways than one. Thanks to the 25% PoW consensus mechanism, they can be mined. Alternatively, they can be earned by selling or buying goods and services within the Marmara Credit Loops system.

****

Source:
https://marmara.io/blog/marmara-chain-1/the-ultimate-simplified-guide-to-marmara-chain-and-marmara-credit-loops-54

Unlock the Power of Marmara Blockchain Staking

Investment Strategies for Maximum APR

Welcome to the exciting world of Marmara Blockchain! As a member of the Marmara Team, I am thrilled to share with you essential information on the Marmara Blockchain staking mechanism and investment strategies to help you achieve the best Annual Percentage Rate (APR) possible. In this comprehensive guide, we will explore the innovative staking system of Marmara and provide you with valuable insights on how to maximize your returns.

Marmara Blockchain Staking Mechanism

Marmara Blockchain employs a unique staking system, which is a combination of 25% mining (Proof of Work) and 75% staking (Proof of Stake). The staking aspect is further divided into two parts: Activated Staking and Credit Loop Staking.

1. **Activated Staking:** By locking your coins in activated staking funds, you gain 1x staking power among the 75% Proof of Stake. Activated staking accounts for 25% of the total staking rewards. The coins can be unlocked at any time, offering you greater flexibility in managing your investments.
2. **Credit Loop Staking:** When you lock your coins in credit loops, you receive 3x staking power. This means that 75% of the total staking rewards come from credit loops. The staking rewards generated in this category are significantly higher than those from activated staking.
3. **Investment Strategies for Optimal APR:** To maximize your APR, you need to consider four critical parameters: total coins locked in activated staking, total coins locked in credit loops within the whole chain, and the number of your coins locked in activated and credit loops. Here are three investment strategies to help you make the most out of your Marmara Blockchain investments:
    - **Analyze the Blockchain Parameters:** Keep an eye on the total coins locked in both activated staking and credit loops within the Marmara Blockchain. This information can be obtained from the Marmara Chain Explorer site. By regularly monitoring these parameters, you can make informed decisions on the best time to lock your coins and in which category.
    - **Diversify Your Investment:** One effective strategy to optimize your APR is by diversifying your investments across both activated staking and credit loops. Allocate a portion of your coins to activated staking, which offers more flexibility due to its unlock feature, and invest the remaining coins in credit loops to take advantage of the higher staking power.
    - **Calculate Your Potential Rewards:** Use the Marmara Staking Calculator to estimate your daily and annual earnings, as well as your annual ROI, based on different investment scenarios. By experimenting with various combinations of coins locked in activated and credit loops, you can identify the optimal allocation that yields the highest APR.

Conclusion

Marmara Blockchain offers a unique and lucrative staking mechanism that allows you to maximize your returns through strategic investment decisions. By understanding the staking system, diversifying your investments, and utilizing the Marmara Staking Calculator, you can unlock the full potential of Marmara Blockchain and achieve an impressive APR on your investments. Happy staking!.

****
Source:
https://marmara.io/blog/marmara-chain-1/unlock-the-power-of-marmara-blockchain-staking-27

Experience the Power of MCL Staking on Marmara Chain: Unlocking Unique Benefits in the Crypto Economy

Marmara Chain, a fork of the Komodo Platform, is revolutionizing the world of staking with its unique and powerful staking mechanism. Offering unparalleled flexibility, Marmara Chain's innovative MCL staking system presents a myriad of advantages over traditional staking platforms. In this article, we'll explore the incredible benefits of staking MCL on Marmara Chain and how it's set to transform the cryptocurrency landscape.

1. No Minimum Staking Requirement: *Freedom to Stake Any Amount*

Unlike other staking platforms like Ethereum, which require a minimum of 32 ETH to participate in staking, **Marmara Chain has no minimum staking requirement.** This means users can stake any amount of MCL, making it accessible and affordable for everyone interested in earning rewards through staking.

2. Flexible Staking: *Unlock Your Coins* Anytime

Marmara Chain's staking system allows users to unlock their staked coins whenever they need them. This flexibility contrasts with other staking platforms where coins are locked for a fixed period, such as six months, with no option to unlock them until the end of the staking term. Marmara Chain's staking system empowers users to retain control over their assets and use them as needed.

3. Marmara Credit Loops: *Boost Your Staking Power by 3x*

Marmara Chain's unique staking mechanism offers two ways to lock coins for staking: activated staking mode and Credit Loops. Activated staking mode provides 1x staking power, while Credit Loops give users an **impressive 3x staking power**. This increased staking power can significantly boost rewards and make MCL staking even more lucrative for participants.

4. Circulation of Staking Coins in the Real Economy

One of the standout features of Marmara Chain's staking system is the ability to circulate staking coins in the real economy. Coins locked in Credit Loops can be transferred or endorsed to other users before their maturity date, similar to post-dated checks or promissory notes. This innovative approach encourages the use of MCL in real-world transactions while still providing staking rewards.

5. 100% Collateralization: *A Secure and Trustworthy System*

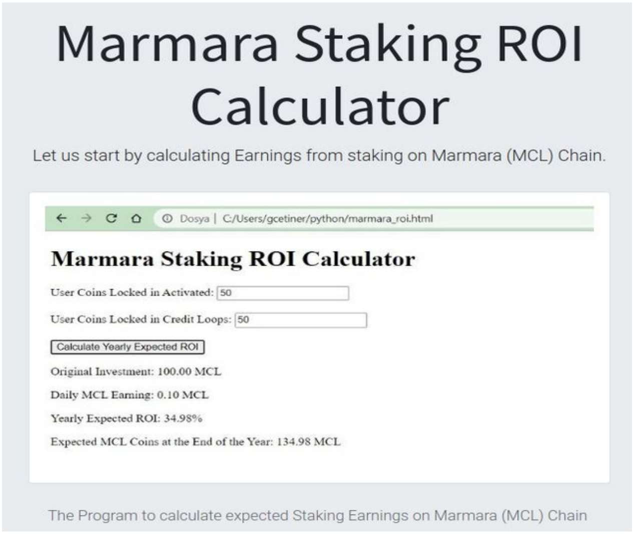
Marmara Credit Loops are fully collateralized on the Marmara blockchain, ensuring a secure and trustworthy staking system. This level of security eliminates the risk of non-redemption often associated with traditional credit tools, such as post-dated checks, and provides peace of mind for MCL stakers.

Conclusion:
Marmara Chain's groundbreaking MCL staking system offers unparalleled flexibility, increased staking power, and the unique ability to circulate staking coins in the real economy. By eliminating minimum staking requirements and allowing users to unlock their assets at any time, Marmara Chain has positioned itself as a leader in the world of cryptocurrency staking. Discover the incredible potential of MCL staking on Marmara Chain and experience the future of decentralized finance today.

****
Source:
https://marmara.io/blog/marmara-chain-1/experience-the-power-of-mcl-staking-on-marmara-chain-21

Why Marmara (MCL) Coin Stands Out

In the rapidly evolving world of cryptocurrencies, passive income opportunities such as staking have become increasingly popular among investors. With countless blockchain platforms offering various staking rewards, it can be challenging to find the right investment for your hard-earned money. This is where Marmara (MCL) Coin, with its impressive 34.98% APR* staking rewards, truly stands out from the competition.
We will compare the staking rewards offered by Marmara Chain to those of other prominent blockchain platforms. We will also discuss why Marmara Chain's unique features make it an ideal choice for investors seeking to maximize their passive income.



Marmara Staking ROI Calculator

Let us start by calculating Earnings from staking on Marmara (MCL) Chain.

Dosya | C:/Users/gcetiner/python/marmara_roi.html

Marmara Staking ROI Calculator

User Coins Locked in Activated: 50

User Coins Locked in Credit Loops: 50

Calculate Yearly Expected ROI

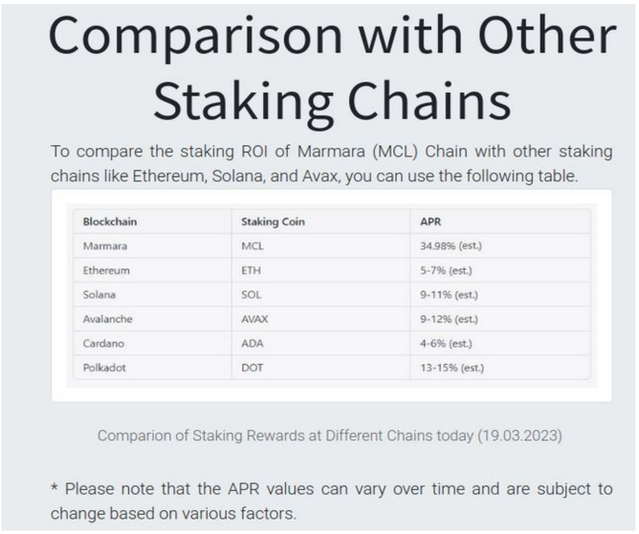
Original Investment: 100.00 MCL

Daily MCL Earning: 0.10 MCL

Yearly Expected ROI: 34.98%

Expected MCL Coins at the End of the Year: 134.98 MCL

The Program to calculate expected Staking Earnings on Marmara (MCL) Chain



Comparison with Other Staking Chains

To compare the staking ROI of Marmara (MCL) Chain with other staking chains like Ethereum, Solana, and Avax, you can use the following table.

Blockchain	Staking Coin	APR
Marmara	MCL	34.98% (est.)
Ethereum	ETH	5-7% (est.)
Solana	SOL	9-11% (est.)
Avalanche	AVAX	9-12% (est.)
Cardano	ADA	4-6% (est.)
Polkadot	DOT	13-15% (est.)

Comparion of Staking Rewards at Different Chains today (19.03.2023)

* Please note that the APR values can vary over time and are subject to change based on various factors.

Marmara Chain's current impressive 34.98% APR is not the only reason why it is an excellent choice for staking. Here are some key factors that contribute to its superiority:

1. **Innovative Credit Loops System:** Marmara Chain's unique Credit Loops system allows users to earn even higher rewards by participating in credit loops that will give you a power of 3x staking.
2. **Decentralization and Security:** Marmara Chain boasts a robust and secure network, ensuring that your staked MCL coins are protected from potential threats.
3. **User-friendly Staking Process:** Staking MCL coins is an easy and straightforward process, making it accessible to both beginners and experienced investors.
4. **The Staking Revolution:** Marmara Chain's staking mechanism comprises 75% Proof of Stake (PoS), meaning the majority of blocks are generated through staking. In comparison to other staking blockchains, Marmara Chain offers several advantages that make it the preferred choice for crypto enthusiasts.
5. **No minimum staking requirement:** Unlike platforms like Ethereum, where a minimum of 32 ETH is required for staking, Marmara Chain allows users to stake any amount of MCL tokens. This inclusive approach enables more people to participate in the staking process without facing financial barriers.
6. **Flexible unlocking:** Marmara Chain offers unmatched flexibility in unlocking staked tokens. Unlike other staking platforms where users must wait for a predetermined period, Marmara Chain allows users to unlock their staked tokens at any time, offering increased liquidity.

7.**Marmara Credit Loops – 3x Staking Power:** Marmara Chain's innovative Credit Loops provide an incredible 3x staking power compared to traditional activated staking. This increased staking power is beneficial for users who participate in the platform's unique Credit Loops system.

8.**Real-world applicability with Credit Loops:** Marmara Credit Loops, inspired by post-dated checks and promissory notes, enable the circulation of staking coins within real economies. Users can transfer or circulate coins locked in Credit Loops contracts before their maturity dates, making MCL a practical solution for real-world transactions.

9.**No need for powerful hardware:** Marmara Chain's staking mechanism eliminates the need for high-performance computing hardware, as its one-minute block time ensures efficient block generation. This accessibility allows more people to participate in staking without worrying about investing in expensive equipment.

10.**The Future of Staking with Marmara Chain:** Marmara Chain's unique staking mechanism, combined with its Marmara Credit Loops, offers an unparalleled experience for crypto enthusiasts. By enabling increased staking power, flexible unlocking, and real-world applicability, Marmara Chain is poised to become a game-changer in the crypto space.

11.**Active Development and Support:** Marmara Chain has a dedicated team of developers and a supportive community, ensuring that the platform continues to grow and improve over time.

12.**Future Growth Potential:** With its innovative features and strong fundamentals, Marmara Chain is well-positioned for continued growth and success in the competitive world of blockchain technology.

Conclusion

In conclusion, Marmara (MCL) Coin offers unparalleled staking rewards when compared to other prominent blockchain platforms. Its impressive 34.98% APR currently, coupled with the innovative Credit Loops system, robust security, user-friendly staking process, and future growth potential, make Marmara Chain an exceptional choice for investors seeking to maximize their passive income.

Don't miss the opportunity to be part of this staking revolution. Join the Marmara Chain community today and experience the benefits of Marmara Credit Loops for yourself.

Don't miss out on this lucrative opportunity to grow your cryptocurrency investments with Marmara (MCL) Coin. Join the thriving Marmara Chain community today and experience the benefits of superior staking rewards for yourself!

****

Source:

https://marmara.io/blog/marmara-chain-1/discover-the-superior-staking-rewards-with-marmara-mcl-coin-a-comprehensive-comparison-23

Marmara Connector

Installing Marmara Connector

Build Marmara Connector from Source

For executable file

Open terminal in file of project. I mean where there is src.

- Test Run

```
fbs run
```

- Create executable file

```
fbs freeze
```

Getting Started with Marmara Connector Development

Preparing the Development Environment

Prerequisites

```
PYTHON3 -> python3.6 is needed for fbs packaging
PyQt5
QT Designer
pip library with latest version
```

Linux

Make sure that the OS is up to date and upgraded:

```
sudo apt-get update
sudo apt-get upgrade
```

Install tools required to build dependencies:

```
sudo apt install git python3 python3-pip python3-venv
```

Clone the repository to a desired location on the local machine:

```
git clone https://github.com/marmarachain/marmara-connector.git
```

For a system-wide PYQT5 and Designer Installation issue the following commands in the terminal:

```
pip3 install pyqt5==5.9.2
sudo apt-get install qttools5-dev-tools
sudo apt-get install python3-pyqt5
sudo apt-get install pyqt5-dev-tools
```

To work with a virtual environment on the project, issue the following commands outside the working directory of the project:

```
python3 -m venv venv
source venv/bin/activate
pip install pip==21.2.4
pip install -r requirement
```

In order to run the project, run mainApp.py under /src/main/python/ directory:

```
python3 mainApp.py
```

Windows

Install the following tools required to build dependencies:

```
git python3 python3-pip python3-venv
```

To install QT5 Designer:

Install the latest version of "pyqt5-tools" using:

```
pip install pyqt5-tools --pre
```

The "designer.exe" will be installed in ...Lib\site-packages\qt5_applications\Qt\bin.

Clone the repository to a desired location on the local machine:

```
git clone https://github.com/marmarachain/marmara-connector.git
```

Change directory to marmara-connector. Create a virtual environment in the current directory:

```
python3 -m venv venv
```

Activate the virtual environment:

```
call venv\scripts\activate.bat
```

Continue the following steps while the virtual environment is active.

Install the required libraries:

```
pip install fbs PyQt5==5.9.2
pip install pip==21.0.1
pip install -r requirement
```

Resources

To setup the Windows OS environment with different python versions, check out here: https://medium.com/swlh/how-to-set-up-a-virtual-environment-with-a-different-python-version-on-windows-10-9900eb0acf9a

Install Marmara Connector from Releases

From Releases, kindly find Assets for the Marmara Connector.

- **Ubuntu Installation**

Download the related .zip file and unzip the file.

To enable the application file to run:

```
chmod +x MarmaraConnector
```

- **Windows Installation**

Download the related .exe file. After the application is installed, run it as an administrator.

Marmara Connector Software Specifications Document

Abbreviations

OS: Operating System
MBC: Marmara Blockchain
UI: User Interface

Definitions

Issuer: The person who first creates a credit in a credit loop. It is the person who forms the first node in the credit loop. The credit may be collateralized 100% or with even zero-collateralization until maturity date of a credit.

Bearer (Holder): The last node in a credit loop is always called bearer (holder). When someone transfers a credit in a loop, that node becomes immediately an endorser.

Endorser: All other nodes that fall between the issuer, which is the first node in the credit loop, and the last node, and transfer the credit to the next node.

Maturity: The time a credit expires. It is measured as blocks in estimation. The block per day is 1440 (60 blocks an hour times 24 hours a day). Suppose a credit is for 100 days, the maturity is 1440x100, i.e. 144,000 blocks.

Settlement: When a credit matures, settlement is made to the holder, the last node in a loop. Settlement may be automatic or manual.

Escrow: Trust based parties for Protocol 2. If EscrowOn is false, then 100% collateralization is used and settlement is automatic. There is no need for escrows in protocol 1 which works as complete trustless version.

Avalist: Avalists support issuer or endorsers with MCL as additional colletarization and can earn with 3x staking with those support. In case of non-redemption, their funds are utilized. Avalists are available only in Protocol 2. The parameter avalcount is always zero for protocol 1.

BlockageAmount: This is a term for protocol 2. An issuer may be asked to put some collateralization by a holder. In that case, the issuer gets benefit of 3x staking in protocol 2.

Dispute Expiry: It is grace period for solving non-redemption problem in credit loops in protocol 2. An issuer may have this time as blocks when creating a credit under protocol 2 without or insufficient collateralization. Before this period expires, an escrow should do all actions according to aggrement with the issuer to solve non-redemption. Otherwise, the escrow is penalized in the system.

Non-Functional System Requirements

- This software is a desktop based GUI application that has distributions for Windows and Linux OS.
- The software is developed in Python using PyQt library as well as Qt Designer for GUI design.
- Bugs, new features, improvements are handled through github issues on Marmara Connector github page.
- Distribution packages are provided under Releases page with details.
- Wiki is constructed for building Connector from source code as well as for guiding

developers to contribute to the open source code base.
- Contributions to the code base are welcomed through pull requests.

Functional System Requirements

- The software is based on Marmara Protocol 1.
- Marmara Connector enables connection to Marmara blockchain running on a local or remote machine/platform.
- The local platform is either Linux, Windows or MacOS based OS.
- The remote platform is Linux based OS.
- The desktop application is used to handle Marmara Credit loop operations as well as wallet operations.

The following parameters are to be handled in the following way at the code base as per decided in Marmara Protocol 1.

- **autosettlement:** AutoSettlement is set to true due to 100% collateralization in Protocol 1.
- **autoinsurance:** Autoinsurance is set to true due to 100% collateralization in Protocol 1.
- **disputeexpires:** Dispute expiry is set to 0 due to 100 collateralization in Protocol 1.
- **EscrowOn:** EscrowOn is set to false due to 100% collateralization in Protocol 1.
- **BlockageAmount:** Blockage amount is set to 0 due to 100 collateralization in Protocol 1.

<u>General constraints/rules</u>

- The users described in this document may be the holder, the issuer or the endorser unless it is explicitly described.
- Block heights that represent maturity block must be converted to datetime data and vice versa.
- For the datetime fields, the earliest date should not be before the commencement of the MBC. The latest datetime should not be further than current datetime.
- For datetime fields, a calender view should be presented in the ui for easy selection of date.
- Every word on the ui must be translatable except for the messages directly received from the MBC.
- Actions and calls made to the MBC and received from the MBC must be logged to a log file wherever applicable and should be reachable by the user.

<u>Server Definition</u>

- The user should be able to create a server record having fields of server name, server user name, ip address and server password.
- The fields server name, server user name and ip address saved by the user should be stored in a file readable, writable by the application. The password must not be stored.
- The user should be able to see (read) all his/her server details from the list and able to update them.
- The user should be able to delete a server from the list.

Business Rules/Constraints/Assumptions (Server Definition)

- In the server page; the server name, server user name, ip address and server password must be required fields and any of them cannot be left blank.
- In case the user wishes to update any of the saved server fields, a message should be popped to warn the user if he/she is sure of the relevant update.
- In case the user wishes to delete a server, a message should be popped to warn the user if he/she is sure of the deletion.
- The server details should be stored in a file even when the user shuts down the GUI application.
- The data directory that contains the server details should stay even if the application is uninstalled or upgraded in the respective host.

Contacts Definition

- The user should be able to create a contact having name, wallet address and pubkey fields.
- The name, wallet address and pubkey fields saved by the user should be stored in a file readable, writable by the application.
- The user should be able to see(read) all his/her contacts from the list and able to update them.
- The user should be able to delete a contact from the list.
- The contacts should be stored in a file even when the user shuts down the GUI application.
- The user should be able to see the name of the person he/she added to the contacts list while making credit loop transactions wherever applicable.

Business Rules/Constraints/Assumptions (Contacts Definition)

- In the contacts page; the name, wallet address and pubkey fields must be required fields and cannot be left blank.
- In case the user updates any of the contacts' fields, a message should be popped to warn the user if he/she is sure of the relevant update.
- In case the user wishes to delete a contact, a message should be popped to warn the user if he/she is sure of the deletion.

Installing MBC

- The program needs to check if MBC is installed.
- The user should be able to install MBC to the platform (local or remote) if it doesn't exist.
- In case MBC is installed beforehand, the user may be asked to provide a path to the src folder.
- During installation, a bootstrap option needs to be provided for downloading bootstrap along with information box displaying the relevant info (Bootstrapping is not recommended).
- During the installation, the user needs to be informed about the process and the progress. (Indexing progress etc.)

Wallet

- The UI should implement two methods for creating a wallet: The user should be able to create a wallet address from his/her own seeds or through the use of getnewaddress command in the UI.
- The user should be able to create a pubkey from the UI (launch MBC with normal launch parameters and execute getnewaddress and validateaddress commands). The errors

produced at the backend should be made available to the user at the UI.

- The UI should enable inputting 12 seed words to create private key from one’s own seeds (convertpassphrase). Warning should be displayed such that the user doesn’t include special characters in those words. The UI should return the agamapassphrase, address, pubkey and wif.
- The user should be able to add a private key to the wallet (importprivkey). The resulting wallet address and the pubkey (after validateaddress command) should be displayed to the user.
- The user should be able to reveal the private key corresponding to an address in his/her wallet.

Setting pubkey and Running/Stopping MBC

- The user should be able to run the MBC node with a specific pubkey. From a list of wallet addresses and pubkeys belonging to the user, the user should be able to set a pubkey and start the MBC with it. Once the MBC node is started with a selected pubkey, an information could be displayed to the user.
- The user should be provided with an option to start the chain with reindexing.
- The user should be able to start the MBC node with a rescan option.
- The user should be able to stop the MBC node. An information could be displayed to the user informing him/her.

Reaching Details about Node and Wallet in MBC

- The user should be able to see and refresh his/her node's Marmarachain status, sync status, current difficult, current block height, longest chain, connections and the latest refresh time from the interface.
- The user should be able to see the pubkey and wallet balances (the normal amount, the activated amount, total normal amount and total activated amount) for his/her node.
- The user should be able to see the total normal amount and total activated amount available in his/her wallet.

Business Rules/Constraints/Assumptions (Setting pubkey and Running/Stopping MBC)

- The UI should not allow setting a pubkey that is short in character length. (The software should not create a means for the user to set a wrong pubkey while starting the MBC.)
- Reindexing option should be defaulted to False.
- Rescanning option should be defaulted to False.

Activating and Deactivating Coin

- The user should be able to activate (lock) and deactivate(unlock) coins from the UI. Rule: The user should be asked for a confirmation of the relevant operation.

Sending Coins to an Address

- The user should be able to send coins directly to an address by inputting the address.
  - The user should be able to send coins by selecting from his/her contact list.
  - The user should be able to generate the QR Code of his/her own wallet address to make a coin receive request.

Business Rules/Constraints/Assumptions (Sending Coins to an Address)

- A warning may be provided such that the user doesn't send coins to an Activated Address. The user should be noted that this action burns coins.

- The user may be informed that a transaction fee is deducted for the transaction being made from one's Normal Amount.

Staking/Mining Mode in MBC

- There are two modes of the MBC node, namely staking and mining.
- The UI should display the current mode of the node (getgenerate).
- The user should be able to change the mode of the MBC node.
- If the user wishes to change the mode of the node to staking, an information could be displayed that warns the user that there should be an activated amount in the wallet for staking.
- If the user wishes to change the mode of the node to mining, the UI should ask the user to input the number of cores to be used for mining. There should be an option to choose the max number of threads.

Credit Loops Operations

Incoming loop requests

- The user should be able to review and confirm the incoming credit loop requests for a given day-time. Rule: The default could be the last 24 hours. If default is not selected, the user shall be able to select a datetime from the calender.
- The user should be able to review the credit loop transfer requests.
- The txid, amount, maturity, receiver pubkey fields should be available for reviewing the incoming loops requests as well as loop transfer requests.
- The user should be asked for confirming the respective transaction. The respective result should be presented in the ui. (txid for confirmation or transaction abort message in case of dismissal).

Making Credit loop requests

- The holder (the one selling the product/service) should be able to request for a credit from the issuer (the one paying the product/service) by entering the amount, sender pubkey, maturity in datetime. The holder shall be able to select the sender pubkey (issuer's pubkey) from his/her list of contacts or insert the pubkey manually.
- A confirmation message should be returned to the holder for confirming the credit loop. The resulting txid should be returned to the holder in case he/she approves the transaction. Otherwise, a message needs to be displayed for the transaction that has been aborted.

Transactions and Loop Queries

- The user shall be able to see his/her own transactions (txid) on the explorer site between his/her choice of selected datetime fields.
- The user should be able query the whole MBC for the active loops he/she has and view the loop address, amount in that loop address, amount in active loops, amount in closed loops, number of pending and number of closed loops.
- The user should be able to query a txid to display the details of a credit loop between the issuer and the holder (the amount with currency, the batonid, the maturity datetime, the issuer pubkey).

Packaging Guide for Marmara Connector for Different OS

For packaging the Qt Gui Application, fbs is used.

Prerequisites

```
PYTHON3 -> python3.6 is needed for fbs packaging
PyQt5
pip library with latest version
```

Guidelines for Linux Packaging

Make sure that the OS is up to date and upgraded:

```
sudo apt-get update
sudo apt-get upgrade
```

Install tools required to build dependencies:

```
sudo apt install git python3 python3-pip python3-venv
```

Clone the repository to a desired location on the local machine and make sure the branch is set to master:

```
git clone https://github.com/marmarachain/marmara-connector.git
cd marmara-connector
git checkout master
```

Check the python3 version. The free version of fbs only supports Python 3.5 and 3.6. So, make sure to switch to python 3.6 for fbs packaging.

```
python3 --version
```

Create a virtual environment in the current directory with python3.6:

```
python3 -m venv venv
```

Activate the virtual environment:

```
source venv/bin/activate
```

Install the required libraries:

```
sudo apt-get install qttools5-dev-tools
sudo apt-get install python3-pyqt5
sudo apt-get install pyqt5-dev-tools
pip install wheel
pip install pip==21.2.4
pip3 install pyqt5==5.15.4
pip install -r requirements
```

To run the basic PyQt application from source, execute the following command:

```
fbs run
```

To turn the app's source code into a standalone executable, run the following command:

```
fbs freeze
```

This creates the folder target/ which then can be copied to any other computer (with the same OS as yours) to run the app there.

To create an installer on Linux, the following command is used:

```
fbs installer
```

However, the installer command requires that you have fpm. One can follow these instructions in here to install it and use the installer command to create the .deb executable file. To install fpm:

```
sudo apt-get install ruby ruby-dev rubygems build-essential
gem install --no-document fpm
```

To make sure fpm is installed correctly, try running the following command:

```
fpm --version
```

Now, running the fbs installer command would create the .deb executable Linux installer.

Guidelines for Windows Packaging

Install the following tools required to build dependencies:

```
git python3 python3-pip python3-venv
```

Open the command prompt with admin privileges and clone the repository to a desired location on the local machine:

```
git clone https://github.com/marmarachain/marmara-connector.git
cd marmara-connector
```

Check the branch and make sure the branch is set to master:

```
git branch
git checkout master
```

Check the python version:

```
python --version
```

The free version of fbs only supports Python 3.5 and 3.6. So, make sure to switch to pyhon 3.6 for fbs packaging. For this refer to the resource provided in 1.

Place the .dll files under C:\Windows\System32 folder.

Create a virtual environment in the current directory:

```
python36 -m venv venv
```

Activate the virtual environment:

```
call venv\scripts\activate.bat
```

Continue the following steps while the virtual environment is active:

Install the following libraries:

```
python -m pip install --upgrade pip
python -m pip install pywin32
pip install wheel
pip install PyQt5
pip install pyqt5-tools
pip install pip==21.3.1
pip3 install pyqt5==5.15.4
pip install -r requirements
```

To run the basic PyQt application from source, execute the following command:

```
fbs run
```

Use the following command to turn the app's source code into a standalone executable:

```
fbs freeze
```

This creates the folder target/. You can copy this directory to any other computer (with the same OS as yours) and run the app there. Before using the installer command on Windows, please install NSIS. Then, create an installer by using the following command.

```
fbs installer
```

This creates the installer directory. Change directory to installer, right click and compile the installer.nsi with NSIS. This creates the MarmaraConnectorSetup.exe file under the target directory.

Using QT Designer to Build Marmara Connector Front End

Having installed Qt Designer, launch the application and open the **"guidesigner.ui"** file in case you want to improve/change the design.

The desired development can be made from the opened QT Designer screen.

After the design part is finished, the .ui file should be converted to the python file.

Convert code ui to python:

```
pyuic5 -x guiDesigner.ui -o guiDesign.py
```

Then, a python file named **"guiDesign.py"** would be created.

Open the main**App.py** file with any IDE of your choice.

In the mainApp.py file, the file named **"guiDesign.py"** is imported. **"Ui_MainWindow"** is extend to the class.

```
from guiDesign import Ui_MainWindow  # Import

class MainClassGUI(QMainWindow, Ui_MainWindow): #Extend

def __init__(self, parent=None):
    super(MainClassGUI, self).__init__(parent)
    self.setupUi(self)
```

In this way, we can reach every feature of the GUI design.
We can try our project with the command below.

```
python3 mainApp.py
```

****
Sources:
https://github.com/marmarachain/marmara-connector/wiki/
https://github.com/marmarachain/marmara-connector/wiki/Packaging-Guide-for-Marmara-Connector-for-different-OS
https://github.com/marmarachain/marmara-connector/wiki/Marmara-Connector-Software-Specifications-Document
https://github.com/marmarachain/marmara-connector/wiki/Using-QT-Designer-to-build-Marmara-Connector-Front-End
https://github.com/mherrmann/fbs-tutorial
https://www.learnpyqt.com/tutorials/packaging-pyqt5-apps-fbs/

MCL Concise Beginner's Guide

1) Download and install Marmara Connector. The installation process will automatically help you synchronize the chain. (When synchronization occurs, the connector will turn green against "Chain Status" and "Chain Synchronous", this process may take up to 1-2 days depending on different factors. You can observe how many more days of blocks will be loaded against "Chain Synchronous". The connector shows not how many days are left, It gives information about how many days of blocks are left.) You can check what action has been taken in the Connector via notifications in the lower left corner.

2) Opening staking and mining without synchronization may cause the blocks to become corrupted. Also, obtaining a wallet address without synchronization will slow down synchronization. After synchronization, you can add an address or addresses with the “Add New Address” option in the “Chain” tab. Then stop the chain and start it with one of the wallet addresses you received.

3) Buy MCL and send it to one of your Marmara Connector wallet addresses.

4) Lock the MCLs you send to your Connector address in parts with the “Lock from Normal Balance” option from the Wallet tab. Congratulations! You started 1x staking with the MCLs you locked.

5) With the "Incoming Loop Requests" in the Credit Loops tab, you can see the credit loop requests you receive and approve the ones you wish. After this process, you will start staking with 3x power. You can also submit your loop requests with the “Request Loop” option in the Credit Loops tab, the MCL amount in your approved requests will be staking 3x, as long as you are a holder. You can use the MCLs you hold in the loops by approving any of the transfer requests you receive with the "Credit Loops-Incoming Credit Requests-Credit Loops Transfer Request" option. However, in that case, 3x staking power will be transferred to the new owner. You can track your staking earnings by selecting and searching the date range you want to see from the "Statistics-Earnings Statistics" tab.

6) If you do not want to participate in the cycles and just want to own or accumulate MCL, you can do this by having an MCL address through Komodo Wallet. Another alternative is the “Zincir Market” application, which is still in the testing phase, where you can shop commission-free and sell your products with MCL.

https://marmara.io/
https://marmara.io/blog/
https://marmara.io/discord
https://www.youtube.com/c/MarmaraCreditLoops
https://github.com/marmarachain
https://t.me/MarmaraChain
https://marmara.io/docs/IMSS2019_WhitePaper_Turkce.pdf
https://coinpaprika.com/tr/converter/mcl-marmara-credit-loops/try-turkish-lira/1/
https://www.coingecko.com/en/coins/marmara-credit-loops
Where to Buy MCL: Kriptrade.com Xeggex.com https://atomicdex.io/ https://safe.trade/
https://nonkyc.io/

Transactions on Cryptocurrency Exchanges and Obtaining MCL

There are five exchanges where you can trade MCL:
Kriptrade.com Xeggex.com
https://atomicdex.io/en/downloads/
Safe.trade Nonkyc.io

You can perform MCL-TL exchange on the Kriptrade.com exchange. This is the easiest way.
Since there is no direct exchange between TL and MCL in other exchanges yet,
We will try to explain other ways that can be followed. First, it is necessary to exchange TL with other intermediary crypto units (BTC, USDT, LTC, etc.) and transfer it to the exchanges where we can purchase MCL. There are alternatives such as Binance Turkey, BTCTurk and others for crypto purchases with TL. Identity confirmation is required to transfer crypto money at the exchange where you buy crypto with TL. Once you have your identity confirmed at the exchanges and completed the waiting period, if any, for the transfer, you will be able to make the transfer.
For membership procedures, you can check out these relevant pages:

For trbinance.com (Binance Türkiye) :
https://www.trbinance.com/tr/support/faq/account-functions-guides/727a8a3f2e0f4dffa6630bc9b5e2dda2
https://www.trbinance.com/tr/support/faq/kyc/18f81821e61b4bbbb8e33ba0b6ff07d6

For binance.com (Binance global) :
https://www.binance.com/tr/support/faq/binance-uygulamas%C4%B1nda-nas%C4%B1l-kay%C4%B1t-olabilirsiniz-360042718372
https://www.binance.com/tr/support/faq/kimlik-do%C4%9Frulamas%C4%B1-nas%C4%B1l-tamamlan%C4%B1r-360027287111

For btcturk.com:
https://bilgibankasi.btcturk.com/hc/tr/articles/360011741098-BtcTurk-e-Nas%C4%B1l-%C3%9Cye-Olabilirim-
https://bilgibankasi.btcturk.com/hc/tr/articles/360011678437-BtcTurk-C%C3%BCzdan-Aktivasyonu

We added these three exchanges to explain how the system works. After you understand how transactions are carried out in these exchanges, you can check for yourself whether there are more suitable opportunities in other "TL - cryptocurrency" exchanges.

When you send TL to your account on cryptocurrency exchanges, the money appears in your account as it is. (Of course, unless your bank makes a deduction) (This is the case for Binance TR and BTCTurk, we can say that the situation is generally the same, at least at the time of writing this article*) You cannot make purchases with TL from Binance global, that will take you to Binance Turkey (TRBinance). directs for this process.

TL deposit-withdrawal pages:
https://www.trbinance.com/blog/support-center-guides/binance-tr-hesab%C4%B1n%C4%B1za-nas%C4%B1l-t%C3%BCrk-liras%C4%B1-yat%C4%B1r%C4%B1l%C4%B1r%3F-82189e6491d9486fb04cf50588657106
https://www.trbinance.com/fees/schedule
https://kripto.btcturk.com/yardim/yatirma-cekme

https://kripto.btcturk.com/yardim/limitler-ve-kurallar

It is not possible to buy and sell "cryptocurrency - TL" with balances below 100 TL on BTCTurk. In TRBinance, the minimum order amount for "TL-crypto" transactions is 10 TL. (on 26.11.2023)

Trading rules:
https://www.trbinance.com/trade-rule
https://bilgibankasi.pro.btcturk.com/hc/tr/articles/360012003838-Minimum-Al%C4%B1m-Sat%C4%B1m-Limitleri-Nelerdir-

You should keep in mind that after the TL appears in your account, exchanges charge a certain commission for the TL - cryptocurrency exchange and the purchases and sales of cryptocurrencies among themselves. Since these commissions can be updated, it is useful to follow the rates on the relevant pages of the exchange you are trading on:
https://www.trbinance.com/fees/schedule
https://kripto.btcturk.com/yardim/komisyonlar
https://www.binance.com/tr/fee/trading

Pairing your Binance and TrBinance accounts allows you to receive some units such as KMD that are not available on TRBinance. With Binance transfer, you can transfer crypto currencies to your Binance global accounts free of charge, and in the same way, you can withdraw your global crypto savings to your TRBinance accounts.
How to Match Binance and Binance TR Account? :
https://www.trbinance.com/blog/5/0b697802eff64f2d800f4cd2abb7b0e6

After purchasing cryptocurrency with TL, you can make the transfer for MCL purchase. (Of course, if you have completed the waiting periods for cryptocurrency transfers in the exchanges, if any) You can make transfers through the blockchain networks supported by both exchanges you will make the transaction. It will be more economical to make transactions through networks with low withdrawal fees. However, before making transactions with networks with low withdrawal fees, you should carefully check whether you can reach the cryptocurrency you want to buy MCL or similar with this cryptocurrency on the opposite exchange. For example, if you transfer LTC to the safe.trade exchange, you can transfer to the LTC-BTC & LTC-USDT markets and the MCL-BTC & MCL-USDT markets. It is useful to check in advance whether the exchanges you plan to switch to are active (whether there is an opportunity to buy or sell at an affordable price without waiting too long).

https://www.binance.com/tr/fee/cryptoFee
https://kripto.btcturk.com/yardim/komisyonlar
https://kripto.btcturk.com/yardim/limitler-ve-kurallar

After checking the withdrawal fee you will pay for the transfer, the minimum withdrawal fee you can make (it is written on the withdrawal page, you will receive a warning if you try to send it below), and whether the other exchange has that network, you should also check the minimum deposit fee in the exchange you will make the transfer to. Unfortunately, stock exchanges do not pay much attention to tears for amounts below the minimum deposit amount. Relevant links of the stock exchanges where you can make these checks in MCL trading:

https://nonkyc.io/fees
https://safe.trade/fees
Additionally, for final checks, you can find information not included in these links on the deposit and withdrawal pages of the relevant coins.

In Komodo Wallet, you can see the advantages that decentralized platforms provide to users. No transaction fees are charged from market makers in buying-selling transactions. However, buyers are charged a 0.13% fee. For buyers using Komodo (KMD), the transaction fee is discounted by 10%, making it only 0.117%. Decentralized platforms do not charge any withdrawal fees but do charge network costs that are paid to the validator of the relevant blockchain.

To make a comparison for buying and selling transactions, Binance receives 0.1% from the buyer and market maker (75 per hundred thousand if commission payment approval is given with BNB); 0.08% from market maker and 0.12% from buyer at BTCTurk; While there is no commission for the market maker in Komodo Wallet, it is 0.13% for the buyer (0.117% if they hold KMD); Safe.trade 0.1% from taker and market maker (was before server upgrade-interface change); Nonkyc.io charges a commission of 0.2% from buyers and market makers (discounts are provided for those who hold NonKYC Tokens, and there is also a 25% discount if commission payment is made with nonkyc).

When we compare withdrawal fees, Komodo Wallet, which is a decentralized platform, appears to be ahead by far as it only reflects network costs to users. Nonkyc.io has set the withdrawal fee as 1 USD for all units, and the advantage of nonkyc.io is the commission income it provides to users with its liquidity pools. Binance, BtcTurk and safe.trade add withdrawal fees to the transfer fees at varying rates. You can follow the current fees from the links we shared in the article.

Let's give an example of some coins with low withdrawal fees on Binance:
LTC (via Litecoin network): 0.001 LTC (Approx. 2 TL)
ETC (via Ethereum Classic network): 0.008 ETC (Approx. 4.50 TL)
RVN (via Ravencoin network): 1 RVN (About 52 cents)
AVAX (via Avalanche network): 0.008 AVAX (Approx. 4.95 TL)
BNB (with BNB Smart chain - bep20): 0.0005 BNB (Approximately 3.37 TL)
BTC (with BNB Smart chain - bep20): 0.0000074 BTC (Approx. 8.07 TL)
USDT(with BNB Smart chain - bep20): 0.3 USDT (Approximately 8.67 TL)
KMD (with Smart chain): 0.002 KMD (Approximately 1.5 cents)

You can withdraw with BTC for free at BTCTurk. However, the minimum withdrawal amount is 0.0005 BTC (Approximately 543 TL). The minimum BTC withdrawal amount with Binance BNB Smart chain - bep20 is 0.000015 BTC. (Approximately 16.36 TL) Therefore, BTCTurk can be preferred for BTC transfers over 550 TL, and Binance can be preferred for lower amounts.

Currently, Binance seems to be more advantageous in sending USDT with the BNB Smart chain (bep20) network. (For exchanges where you can buy MCL)

For AVAX (for Avalanche c-chain and Avalanche x-chain networks) withdrawal, BTCTurk stands out with its withdrawal fee of 0.005 Avax (approximately 3.07 TL).

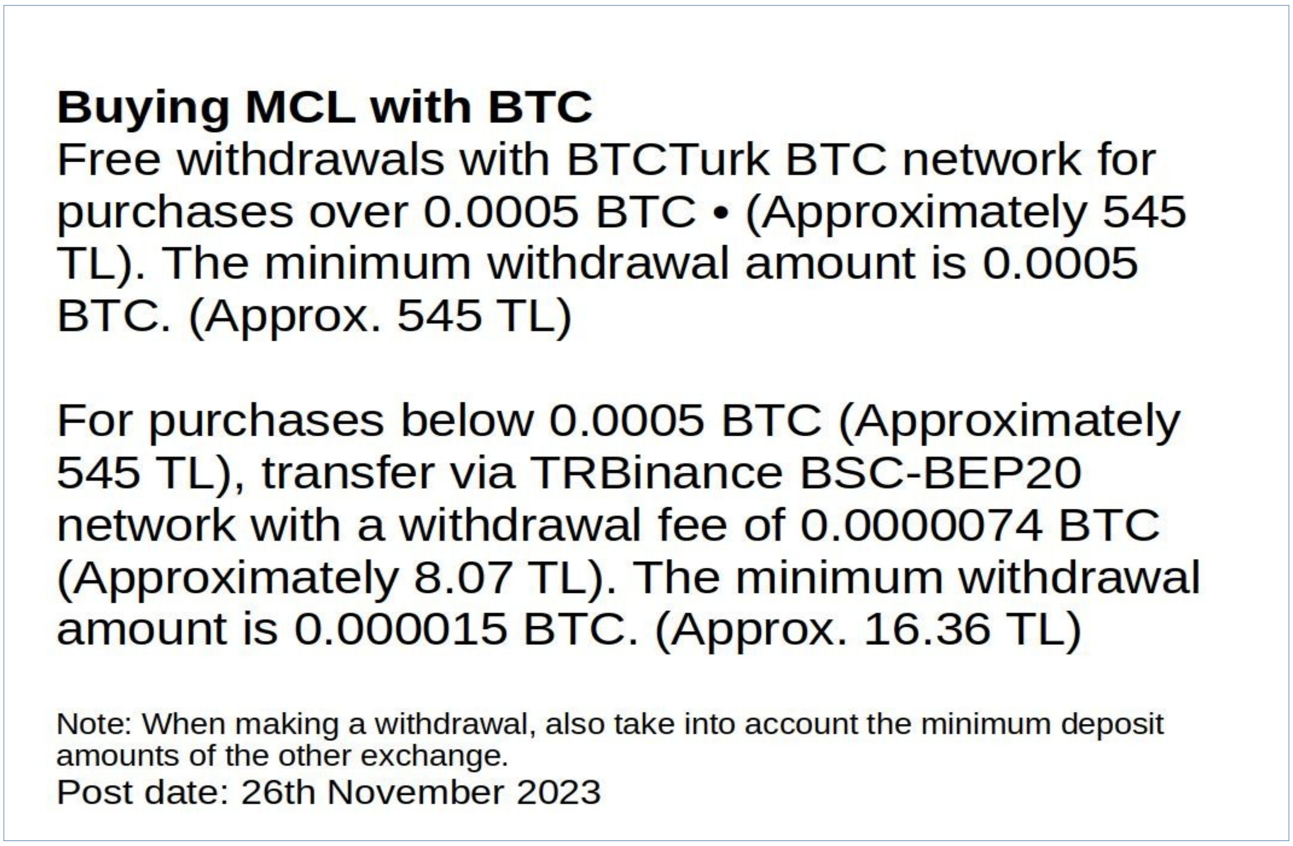
Another alternative is to switch to other exchanges by purchasing RVN from TRBinance. MCL purchase can be made by converting RVN into one of the units traded with MCL on the relevant stock exchange.

After switching to Binance global with a free Binance transfer by purchasing USDT or BTC from TRBinance, you can purchase MCL with Komodo Wallet by purchasing and transferring KMD.

LTC is one of the important coins that allows economical transfer between exchanges with low transfer fees.

MCL is a coin with its own intelligent chain and extremely low transfer fees. The withdrawal fee on Komodo Wallet is only 0.00001 MCL. (Approximately 0.000005 TL) Safe.trade applies a withdrawal fee of 0.01 MCL (Approximately 0.005 TL). As you know, Nonkyc.io applies a standard fee to all coins.

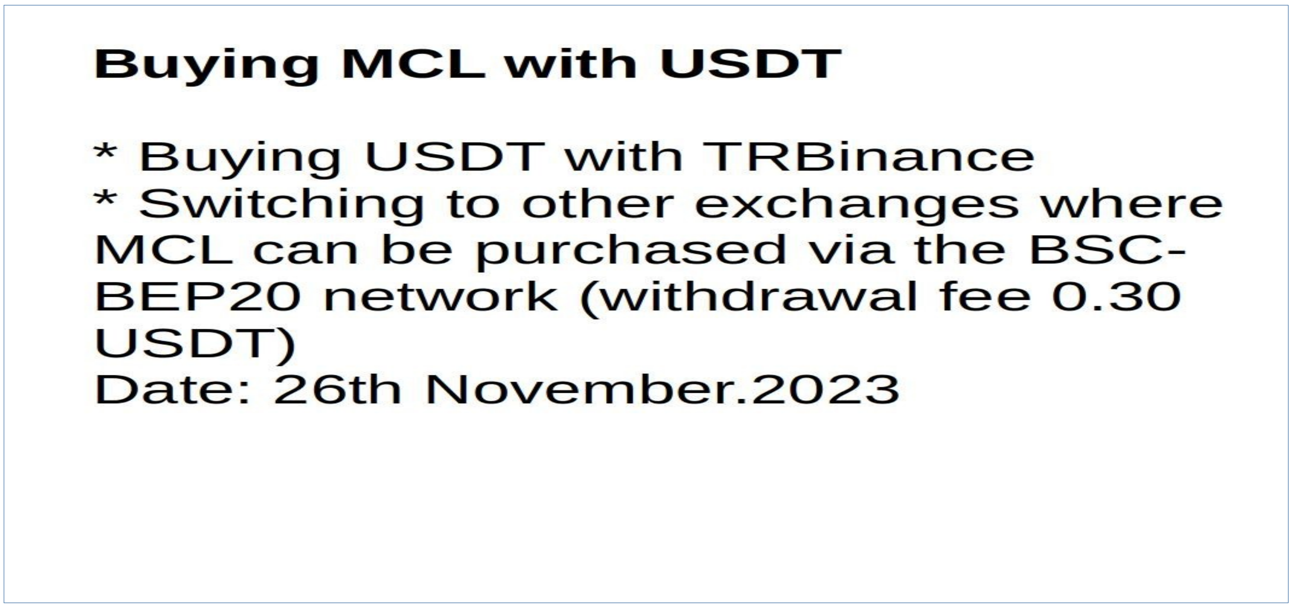
As a result, if you understand how the system works among blockchain networks, you can determine your own path and review different alternatives. You can progress by making advantageous purchases and sales across networks.
We made the center kick and passed the pass to you, now the ball is in your court :)



Buying MCL with BTC
Free withdrawals with BTCTurk BTC network for purchases over 0.0005 BTC • (Approximately 545 TL). The minimum withdrawal amount is 0.0005 BTC. (Approx. 545 TL)

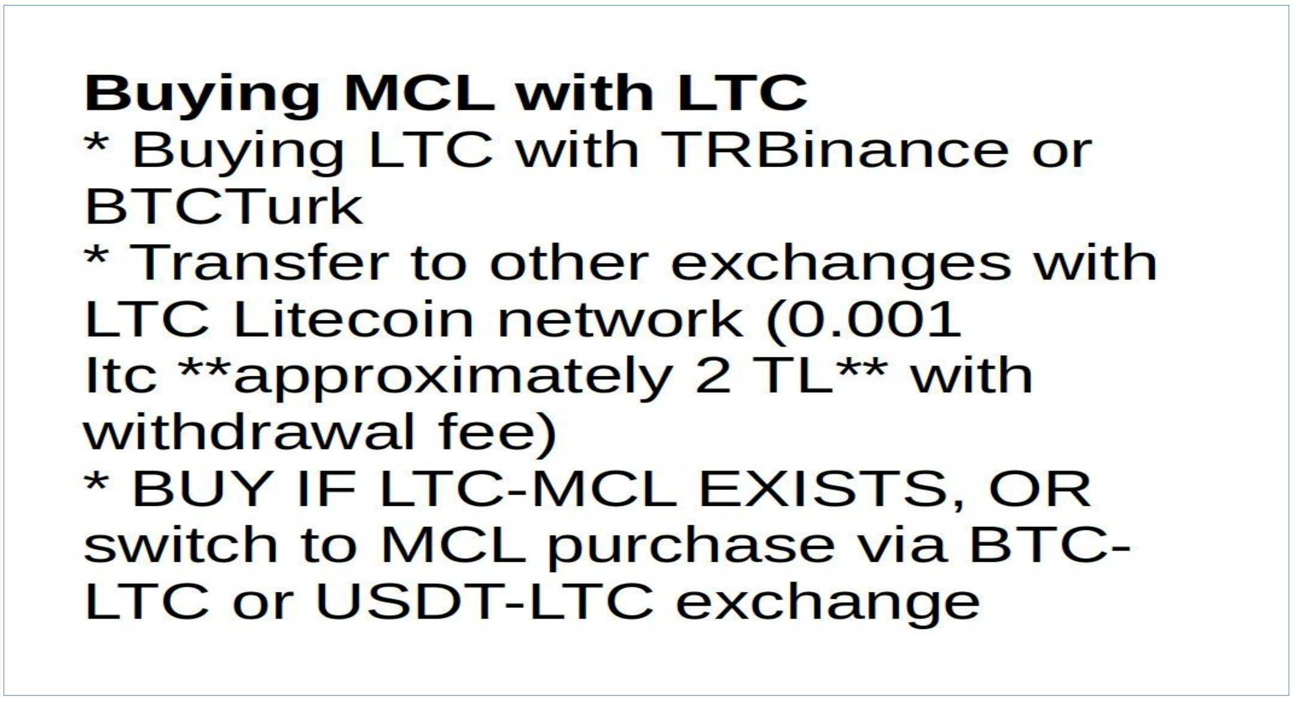
For purchases below 0.0005 BTC (Approximately 545 TL), transfer via TRBinance BSC-BEP20 network with a withdrawal fee of 0.0000074 BTC (Approximately 8.07 TL). The minimum withdrawal amount is 0.000015 BTC. (Approx. 16.36 TL)

Note: When making a withdrawal, also take into account the minimum deposit amounts of the other exchange.
Post date: 26th November 2023



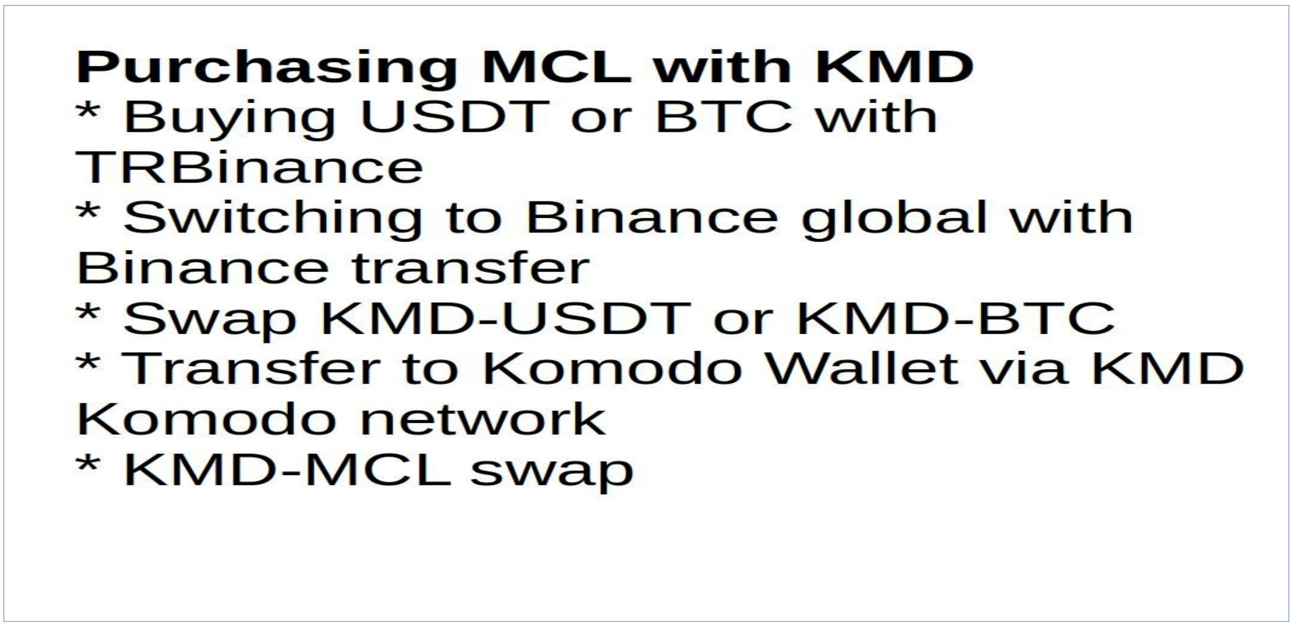
Buying MCL with USDT

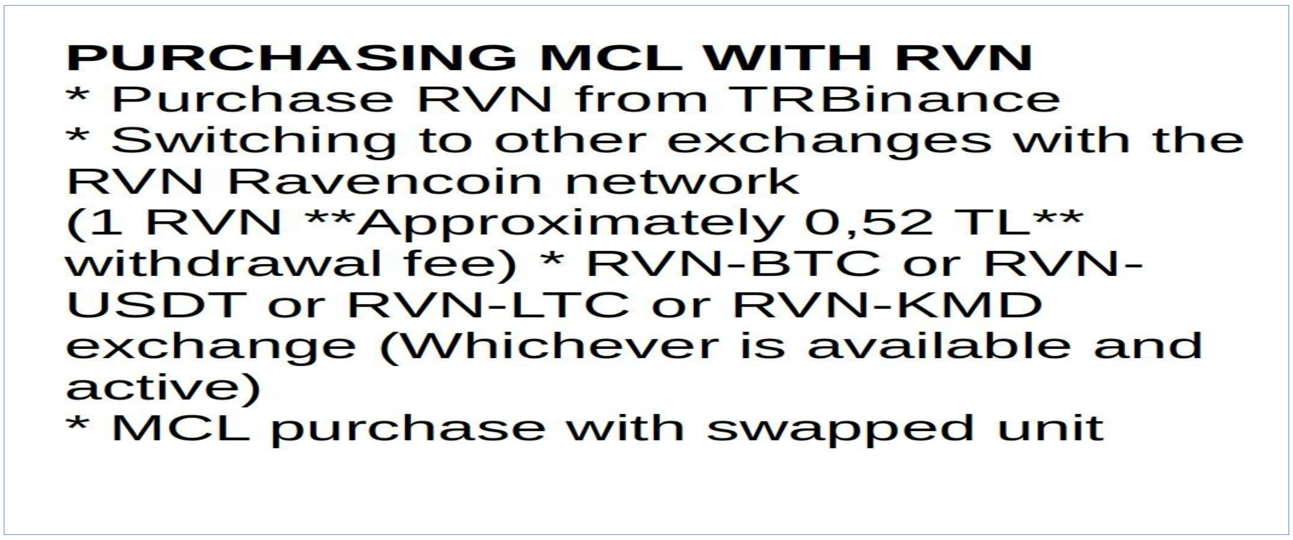
* Buying USDT with TRBinance
* Switching to other exchanges where MCL can be purchased via the BSC-BEP20 network (withdrawal fee 0.30 USDT)
Date: 26th November.2023



Buying MCL with LTC

* Buying LTC with TRBinance or BTCTurk
* Transfer to other exchanges with LTC Litecoin network (0.001 ltc **approximately 2 TL** with withdrawal fee)
* BUY IF LTC-MCL EXISTS, OR switch to MCL purchase via BTC-LTC or USDT-LTC exchange


Purchasing MCL with KMD
* Buying USDT or BTC with TRBinance
* Switching to Binance global with Binance transfer
* Swap KMD-USDT or KMD-BTC
* Transfer to Komodo Wallet via KMD Komodo network
* KMD-MCL swap


PURCHASING MCL WITH RVN
* Purchase RVN from TRBinance
* Switching to other exchanges with the RVN Ravencoin network
(1 RVN **Approximately 0,52 TL** withdrawal fee) * RVN-BTC or RVN-USDT or RVN-LTC or RVN-KMD exchange (Whichever is available and active)
* MCL purchase with swapped unit

Note: The calculations in this article were made according to the exchange rates on November 26, 2023.

*This article was prepared while the law on cryptocurrencies was in draft form.

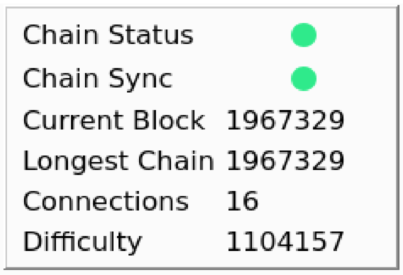
Answers to Frequently Asked Questions

The simplest way to install a Marmara Chain is to install using the latest version of Marmara Connector. Marmara Connector will check whether the Marmara Chain is installed in your system and if not, it will perform the installation process if you give approval. If you have installed the Marmara Chain on your operating system before installing the Marmara Connector, it will connect and enable you to easily perform your MCL operations through the graphical interface. If you encounter any problems during the installation process, you can get help and support from discord or telegram. Download and install the appropriate package for your operating system from https://github.com/marmarachain/marmara-connector/releases

In order to use Marmara credit loops, the user is expected to have a pubkey and start the chain with this pubkey. It is necessary to has a pubkey in order for the coins earned through mining and staking to go to an account. If the chain is not started with pubkey, mining and staking transactions will be wasted.

During the specified transaction period, credit loops can be circulated in the economy in exchange for goods and services up to the 1000th node.

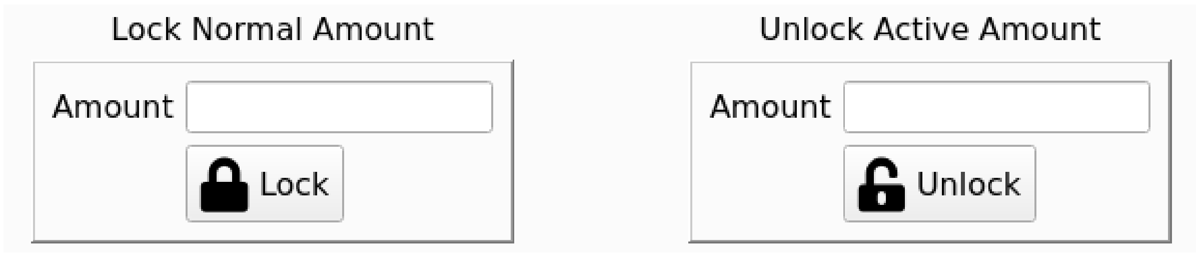
MCL can be moved economically between cryptocurrency exchanges and converted into other values with low transfer fees.



Chain Status	●
Chain Sync	●
Current Block	1967329
Longest Chain	1967329
Connections	16
Difficulty	1104157

The synchronization time of the Marmara chain varies depending on your network connection speed, your computer's processor power, the read-write speed of your drive where the blocks are recorded, and similar factors. Synchronization can be completed in up to 1 day under good conditions, or in 2-3 days under slightly more disadvantageous conditions, but on computers that barely meet the minimum requirements, the synchronization period may take longer. You can monitor the synchronization time via Marmara Connector. In cases where the synchronization time may be long, you can speed up the process with bootstrap. (Bootstrap is not recommended, it is better to perform the synchronization yourself.) When synchronization is achieved in Marmara Connector, it will light green against chain status and chain sync as shown in the picture. Do not open staking or mining without synchronization, you may corrupt the database and have to synchronize again. When synchronized, we recommend creating a wallet address, transferring MCL to your account, then creating loops. It takes more time to find with 1x. Half of the rewards are normal coins and the other half are active coins. Normal coins are shown in odd blocks and active coins are shown in double digits. The contents of single-digit coins appear on the Explorer, while those with double digits are the smart contract addresses of active coins. From the outside it appears as zero on the Explorer, but it is not zero. The power of credit cycles is much greater when staking on Marmara Chain. Sometimes it can even exceed 3x. If everyone is locking in the loop, the loops can drop to just under 3x.

Staking cannot be done with normal coins. You must first lock the amount you want to stake. When locking, it is recommended to lock it in small sections such as 300&500&1000 MCL at a time. Similarly, when transferring to loops, it will be useful to create loops in pieces instead of locking them all at once. Throwing a smaller number of single hooks increases the chances, rather than one large single-hook line at a time. It also speeds up loops. Because this locked payment information is used in loops.



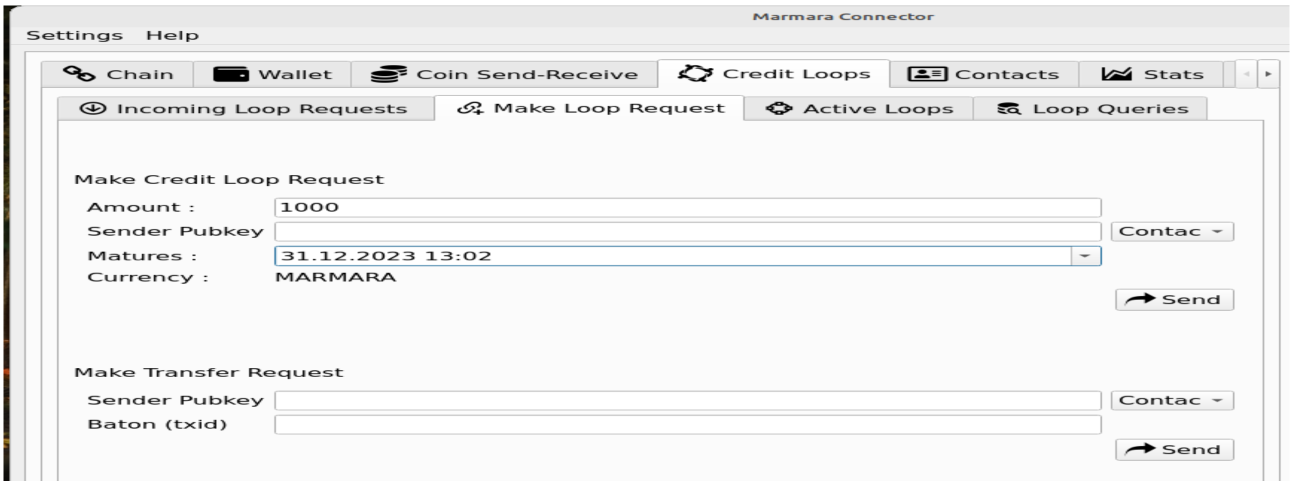
You can easily lock and unlock MCL under the Marmara Connector "Wallet" tab.

You can start making consistent profits 8-10 days after you start staking. We can compare the initial waiting period to a kind of incubation process. But if you lock it in small pieces like 300 & 500 MCL and then include them in loops, you have a chance to find the blocks faster. Under these conditions, if you start with a sufficient amount of MCL, you can find a block even from the first day.

It would be a good idea to take a look at the Zincir.market application which allows you to shop with MCL and you can sell your own products with MCL without commission.

Mining is not recommended unless you have a powerful computer with high processing power. If you are going to do it, it would be better to use a suitable machine for this job. Recommended are Asic miner devices. Instead of mining, you can produce MCL by staking with simpler computers.

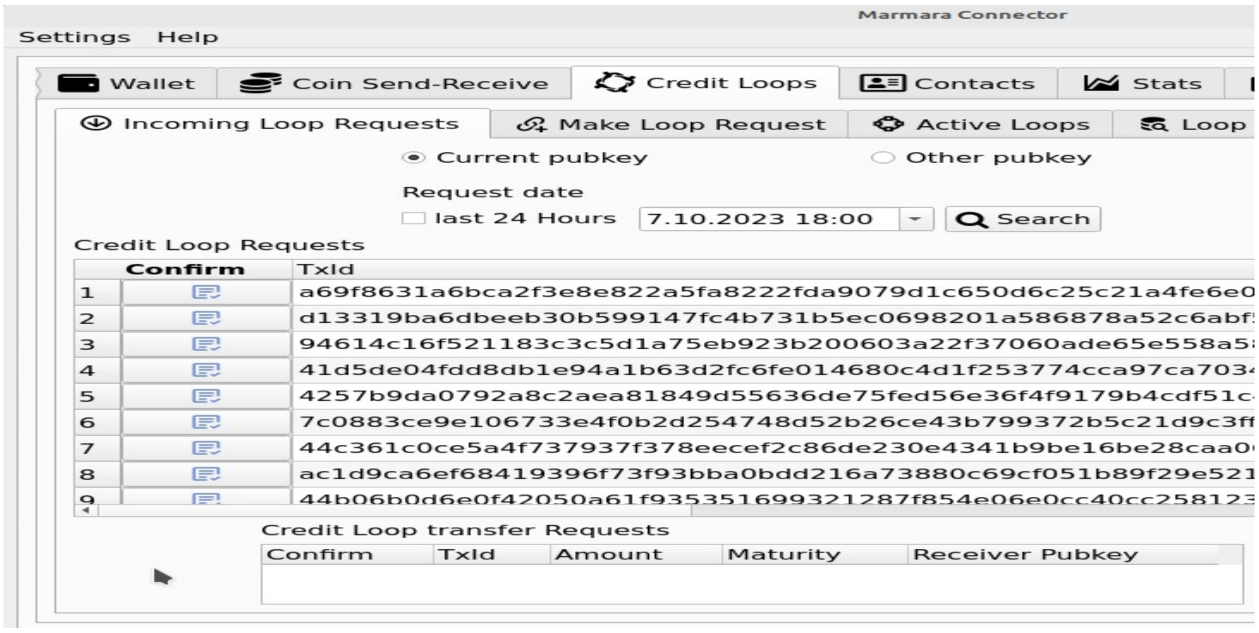
Bootstrap backup is taken daily in the second browser. https://explorer2.marmara.io/bootstrap



Make Credit Loop Request
Amount : 1000
Sender Pubkey
Matures : 31.12.2023 13:02
Currency : MARMARA

Make Transfer Request
Sender Pubkey
Baton (txid)

You can make Marmara credit loop request and transfer request under Marmara Connector “Credit Loops - Make Loop Request” tab.



[✓ Current pubkey] Other pubkey

Request date

☐ last 24 Hours 7.10.2023 18:00 Search

Credit Loop Requests

	Confirm	TxId
1		a69f8631a6bca2f3e8e822a5fa8222fda9079d1c650d6c25c21a4fe6e0
2		d13319ba6dbeeb30b599147fc4b731b5ec0698201a586878a52c6abf
3		94614c16f521183c3c5d1a75eb923b200603a22f37060ade65e558a5
4		41d5de04fdd8db1e94a1b63d2fc6fe014680c4d1f253774cca97ca703
5		4257b9da0792a8c2aea81849d55636de75fed56e36f4f9179b4cdf51c
6		7c0883ce9e106733e4f0b2d254748d52b26ce43b799372b5c21d9c3ff
7		44c361c0ce5a4f737937f378eecef2c86de230e4341b9be16be28caa0
8		ac1d9ca6ef68419396f73f93bba0bdd216a73880c69cf051b89f29e521
9		44b06b0d6e0f42050a61f9353516993212877f854e06e0cc40cc258123

Credit Loop transfer Requests

Confirm	TxId	Amount	Maturity	Receiver Pubkey

You can see and accept easily incoming Marmara credit loop request and transfer requests under Marmara Connector “Credit Loops - Incoming Loop Request” tab.

To see the latest current status in the system, the last block status and the current block status must be synchronized. For example, let's say someone sent you 5 MCLs, you saw it in your account. However, there was a problem in the following days and you started to load the blocks again from scratch. You will not be able to see the 5 MCLs in your account until the corresponding block is uploaded. So what you see on your system is the status of the last block you loaded.

For Ubuntu or Linux Mint: If you install the Marmara chain in advance, you can keep it in the src folder in the marmara folder in your home directory so that the Marmara Connector system can find it (with marmara-cli and marmarad inside the directory)

Related formulation for staking account:
([(MyActivated/TotalActivated)+(MyLoops/TotalInLoops)]*3)*8100

During the installation process with Marmara Connector, you may encounter problems such as curl installation problems or zip files not opening, especially on Windows systems older than Windows 10. In this case, you can get around the problem by unzipping the zip file yourself or installing curl yourself.

You can perform automatic installation on the virtual server with Marmara Connector. There is no need to deal with codes anymore. After selecting the host option instead of local, simply enter details such as IP address and password. Connector does the rest. The chain runs on the server independently of the Connector. The connector is only for connecting and doing control and similar works.

You can create your own staking and investment pool on Marmara Chain with Marmara Connector, which allows you to easily set up the Marmara Blockchain and stake with 3x power by making credit cycles similar to post-dated checks. The new Marmara Connector comes with a gain measurement tool that can make precise measurements to precisely count the gains between two different dates on a pool. The Marmara Investment pool concept is integrated with an application, allowing you to add a self-financing feature to your projects. For example, thanks to integration with the investment pool, it is possible to reset commissions for both buyers and sellers on an e-commerce site. Marmara Chain can likewise be integrated with each project either with a stake-based crowdfunding system or independently. For example, you can integrate the investment pool with a scholarship system. The funds collected in the investment pool can be integrated with 3x staking and used to distribute regular income to scholars. Or it can be integrated with a performance system. Self-financing feature can be provided by integrating with Supply Chain Management or ERP applications.

<u>How to Start Loops from Command Line</u>

Loops are done between two people. So you need someone else to issue or receive. It is easier when you shop with MCL. The receiver will type the following command line:

```
marmarareceive “pubkeyofissuer” MCLamount MARMARA time '{"avalcount":"0"}'
```

Time is set in minutes. As there are 1440 minutes in a day, you have to multiply day number by 1440. If you want to receive 300 MCL in 3 months, type the following command line by changing issuer’s pubkey:

```
marmarareceive “pubkeyofissuer” 300 MARMARA 129600 '{"avalcount":"0"}'
```

This will give you hex, you need to type sendrawtransaction HEXnumber. This step will generate TXID which has to be send to the issuer.

The issuer needs to type the following command line to confirm the loop:

```
marmaraissue "receiver’s pubkey" '{"avalcount":"0", "autosettlement":"true",
"autoinsurance":"true", "disputeexpires":"0", "EscrowOn":"false", "BlockageAmount":"0" }'
TXID provided by the receiver
```

And type sendrawtransaction hexnumber as well.

That’s all ☺

You can find the answers to the different questions you are looking for on Marmara Discord or Marmara Telegram channels.

Links

Official Website:
http://marmara.io/

History:
https://marmara.io/tr/history

Team:
https://marmara.io/tr/aboutus

Wallets:
https://marmara.io/tr/wallet

You can buy MCL from:
Kriptrade.com
Xeggex.com
https://atomicdex.io/
https://safe.trade/
https://nonkyc.io/

Blog:
https://marmara.io/blog/

Whitepaper:
https://marmara.io/docs/IMSS2019_WhitePaper_English.pdf

Block Explorers:
https://explorer.marmara.io/
http://explorer2.marmara.io/
http://explorer3.marmara.io/

Coin Supply:
https://marmara.io/tr/marmara-coin-supply

Staking Info:
http://stake.marmara.io/ (Staking calculator)
https://marmara.io/blog/marmara-chain-1/marmara-staking-calculation-22
https://marmara.io/blog/marmara-chain-1/discover-the-superior-staking-rewards-with-marmara-mcl-coin-a-comprehensive-comparison-23

DEX Stats:
https://mcl.explorer.dexstats.info/

MCL Exchange Rates:
https://coinpaprika.com/tr/converter/mcl-marmara-credit-loops/try-turkish-lira/1/
https://www.coingecko.com/en/coins/marmara-credit-loops

Bootstrap backup is taken daily in https://explorer2.marmara.io/bootstrap
Bootstrap alternative link : https://eu.bootstrap.dexstats.info/MCL-bootstrap.tar.gz

Other Links:
https://marmara.io/discord
https://t.me/MarmaraChain
https://marmarachain-mcl.medium.com/
https://github.com/marmarachain
https://github.com/marmarachain/marmara/releases
https://zincir.market
https://mclmarket.com/
http://www.drcetiner.org/
https://komodoplatform.com/
https://discord.com/invite/4a3J7gAmC5
https://github.com/KomodoPlatform
https://github.com/marmarachain/marmara/wiki
https://github.com/marmarachain/marmara-connector/wiki
https://www.youtube.com/@MarmaraCreditLoops/videos
https://www.youtube.com/@gultekincetiner/videos
https://www.youtube.com/@KomodoPlatformOfficial/videos
https://www.youtube.com/channel/UCBabFGXtvtThqKaYYv5hdVA/videos
https://twitter.com/marmarachain
https://www.instagram.com/marmarachainofficial/
https://www.facebook.com/groups/marmarachain
https://coinpaprika.com/coin/mcl-marmara-credit-loops/
https://www.livecoinwatch.com/exchange/atomicdex
https://cryptorating.eu/whitepapers/Komodo/2018-02-14-Komodo-White-Paper-Full.pdf
https://developers.komodoplatform.com/basic-docs/atomicdex/introduction-to-atomicdex.html
https://scet.berkeley.edu/wp-content/uploads/BlockchainPaper.pdf
https://developers.komodoplatform.com/
http://sanaloba.org/
http://guneskoyu.org/

Important Notice to Users

Marmara Chain

This software is based on Komodo which is based on zcash and is considered experimental and is continously undergoing heavy development. Marmara Credit Loops is experimental and a work-in-progress. Use at your own risk.

At no point of time do the MCL and Komodo Platform developers take any responsibility for any damage out of the usage of this software.

Marmara Connector

Marmara Connector is experimental and a work-in-progress. Use at your own risk.

****

Sources:
https://github.com/marmarachain/marmara/wiki/Important-Notice-to-Users
https://github.com/marmarachain/marmara-connector/wiki/Important-Notice-to-Users

Issue Reporting Format – Marmara Connector

Please report your issues with format given below by submitting them in here:
https://github.com/marmarachain/marmara-connector/issues

```
Impacted versions: .....

Steps to reproduce:
....

Current behavior:
.....

Expected behavior:
....

Video/Screenshot link (optional):
```

License Details

Marmara Chain & Marmara Connector

****

Sources:
https://github.com/marmarachain/marmara/wiki/License-Details
https://github.com/marmarachain/marmara-connector