Civitas Fundamenta

02.28.2021

—

Written by Steve Medley & Matt Hooft

Civitas Fundamenta and the Fundamenta Token

Overview

Civitas Fundamenta launched back in November 2020 with a different goal in mind. We were aiming to be a community governed funding platform but as we were building very early on we realized that Ethereum fees were a growing issue and that a number of projects were looking for solutions. Projects began to talk about migration from Ethereum to other chains and some had an even bigger idea. Why not make our projects cross chain compatible? And thus began the arms race for cross chain compatibility. We saw an up and coming opportunity and decided that we would enter this sector with our own offering of cross chain compatible assets. Through our platform users will be able to take any ERC-20 token and use our bridge “CiviPort” to make their assets cross chain compatible. This is possible through a technique we are calling Digital Asset Teleportation. Civitas Fundamenta will also be wrapping assets like Bitcoin, Litecoin and Monero and creating CiviPort compatible ERC-20 tokens to represent them. This will make assets truly cross chain compatible and will allow users to use their capital on the chain or chains of their choosing in any way they wish.

There will be fees to wrap and unwrap assets as well as a minor fee to use CiviPort. Holders of our governance token FMTA will be able to stake their tokens and receive a share of these fees as well as receive daily rewards in the form of FMTA from staking emissions.

We will also be building investment vehicles to utilize all of our assets. One of these vehicles is called an Asset Tracking Token or ATT. ATT’s represent ownership of a number of underlying assets and are tradeable on any decentralised or centralised exchange.

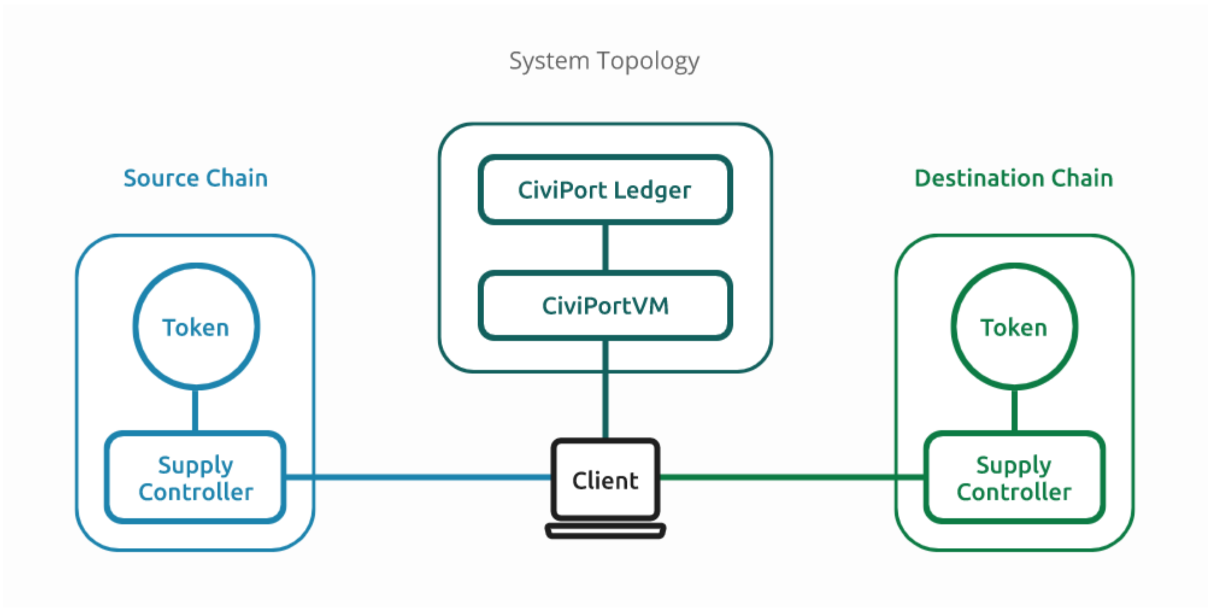
What sets us apart from other cross chain solutions is we are building this with regulatory compliance in mind. Civitas Fundamenta is a registered Money Service Business with FinTRAC (M21335902) and will have a compliance program in place. We believe that there is a need to bridge the divide between compliance and decentralisation and we are looking to fill that need. Users can still interact with our services without having to perform any KYC procedures but they will be limited to $1000 in transaction value per week. We will achieve this by recording IP Addresses and linking them to ethereum and assets accounts and addresses. Users who wish to move more than $1000 are more than welcome to complete KYC and gain access to the system with heightened or no limits. Note that KYC only applies to wrapping and unwrapping assets. CiviPort use will not be subject to KYC or limits.

Alright that’s the ugly stuff out of the way let’s get into the technicals of CiviPort and how it will function.

CiviPort

CiviPort is the name of our bridge that will allow FMTA and eventually all ERC-20 and supported assets to be cross chain compatible. This is achieved by using a technique called asset teleportation. In all honesty the method is quite simple. Users who wish to make an asset cross chain compatible through CiviPort can do so by wrapping their asset or ERC-20 with CiviSwarm (detailed later). Once that is complete and a user wishes to move from one chain to another they initiate a teleport and the CiviPort compatible asset is *dematerialized* (burned) and the teleport data is sent to CiviPortVM and the CiviPort Ledger which is a permissioned Quorum blockchain (https://consensys.net/quorum/). When the user is ready and enough confirmations on the originating chain have passed they can then complete the teleport. CiviPort then uses the data from the CiviPortVM and Ledger to *re-materialize* (mint) the asset on the other chain.

At a macro level, the system works via a withdrawal from the source chain and a deposit to the destination chain. CiviPortVM authenticates these requests by checking they are valid and issuing a deposit token which can then be used to deposit the tokens onto the destination chain. The CiviPortLedger contract works as an on chain storage system for these client requests and serves as a central on-chain storage system to assist CiviPortVM in determining the validity of requests and for auditing purposes so auditing tools can use the ledgers data to track the movement of coins between chains for regulatory compliance. The bridge contract acts as an authorized agent to mint and burn coins on the token contract. the bridge contract only responds to properly formatted messages that have been signed both by the client and CiviPortVM. This ensures that action is only taken on a client's funds if the action is authorised by the client and confirmed by CiviPortVM



The overall process for both a withdrawal and a deposit are essentially the same and can be broken down into a discrete set of steps.

- The client submits a signed request to the server for the withdrawal/deposit action
- The server checks the validity of the request and responds with a signed response message
- The client then co-signs the response message
- The client then submits the message to the bridge contract
- The bridge performs the requested mint/burn action on the attached token contract

The structure of the request and response messages differs between deposits and withdrawals. The next sections will discuss in detail the complete process.

Withdrawal

To withdraw, the client crafts a message body as a byte array 104 bytes long with the following format:

- amount to withdraw (uint256)
- bridge ID (i.e. the identifier of the bridge to withdraw from) (uint64)
- Nonce (randomly generated) (uint256)
- Client address (32 bytes, 12 padding bytes + 20 byte address)

The client then hashes the data with keccak256 and signs the resulting hash. These 2 pieces of information are sent via rpc to CiviPortVM in the following JSON format:

```
{
  "api": {
    "major": 1,
    "minor": 0
  },
  "data": {
    "data": "<hex encoded message body>",
    "sig": "<hex encoded data signature>"
  }
}
```

NOTE: *the api block is used internally by the RPC system and is required for every request*

When the server receives the message, it decomposes the data buffer back to its original inputs. The server also recovers the signing address from the message and signature. The address in the message buffer must match the signing address to proceed. This ensures that request is authorized by the account the tokens will ultimately be burned from.

The server then proceeds to check that the provided bridge ID is a valid identifier and that the client has a sufficient balance in that token contract to process the withdrawal.

The server then queries the ledger to ensure the nonce has not been used previously.

If all these checks pass, the withdrawal data is added to the ledger. The ledger records:

- Nonce
- Withdrawal bridge ID
- Client address
- Amount to withdraw

The server will then hash and sign the original request data and respond to the client with this signature and a copy of the nonce recorded in the ledger:

```
{
  "data": {
    "data": "<Withdrawal nonce>",
    "sig": "<Server signature of request data>"
  },
  "status": "OK",
  "code": 0
}
```

When the client receives the response, the client then submits the message buffer, it's own signature and the server authorization signature to the bridge and pay the network fee for the contract interaction.

The bridge contract will recover the client and server signing addresses from the signatures. It then verifies that the recovered server signature is authorized to perform the withdrawal action on the bridge. If an attempt is made to manipulate the message data after being signed by the server, the recovered address will be incorrect and the transaction will fail.

A second check is done that the client address is the signing address of the message by comparing the recovered address to that in the message. This prevents a scenario where a secondary party could re-sign the message with a different private key after receiving the server response.

A check is then done to ensure the bridge ID in the message matches the correct bridge being called. This prevents sending a signed message to a different bridge in a malicious attempt to withdraw coins from a different bridge.

Finally a second check is done to ensure the bridge has not processed a message with the same nonce before. This prevents multiple submissions of the withdrawal message to the bridge contract.

If all these checks pass, the requested amount is burned from the token contract attached to the bridge and the withdrawal is complete.

Deposit

Once the withdrawal is complete, the client can use the provided nonce to deposit the equivalent amount of tokens on another chain. To do this, the client crafts a data buffer consisting of:

- The ID of the bridge they wish to deposit on (uint64)
- The nonce provided by the withdrawal request (uint256)

The client then signs this data and sends an rpc request to the server in the following format:

```
{
  "api": {
    "major": 1,
    "minor": 0
  },
  "data": {
    "data": "<hex encoded message body>",
    "sig": "<hex encoded data signature>"
  }
}
```

NOTE: *The JSON structure of the deposit message is the same as the withdrawal request*

When the server receives the message, it decomposes the data buffer back to it's original inputs. The server also recovers the signing address from the message and signature.

The server then checks the deposit bridge ID is valid. If this is invalid the server cannot proceed.

It also queries the ledger for a corresponding withdrawal for the nonce. The ledger must have recorded the withdrawal previously to continue. The server will extract the amount and the withdrawing user from this data. The address recovered from the deposit signature must be the same as the address recorded for the withdrawal to proceed. This prevent secondary parties from attempting to claim a deposit.

The server then checks if the deposit has been added to the ledger previously and will fail if a duplicate deposit attempt is detected.

If all checks pass, the server will construct a message in the same format as the original withdraw request, i.e:

- amount to withdraw (recovered from the ledger) (uint256)
- bridge ID (i.e. the identifier of the bridge to deposit to) (uint64)
- Nonce (same as that provided by the client in the deposit request) (uint256)
- Client address (32 bytes, 12 padding bytes + 20 byte address)

The server then hashes and signs this message and returns it to the client in the same format as the withdrawal response:

```
{
  "api": {
    "major": 1,
    "minor": 0
  },
  "data": {
    "data": "<hex encoded message body>",
    "sig": "<hex encoded data signature>"
  }
}
```

When the client receives this message from the server, the client then co-signs the message and submits the message, server signature and client signature to the bridge contract and pays the contract interaction fee.

The same checks are conducted on the bridge contract for a deposit as a withdrawal to ensure the message has not been tampered with and has not been submitted previously. If all checks pass, the requested amount of tokens is minted to the clients account and the swap process is completed.

Auditing and Regulatory Compliance

The system is designed with regulatory compliance and ease of auditing in mind. To this end, the CiviPort ledger contract provides the information required to trace movement of tokens between chains and complements auditing of per-chain token movements.

For a withdrawal the ledger records the following information:

- Nonce
- Bridge ID
- User address
- Number of tokens

For each deposit, the ledger records:

- Nonce
- Bridge ID

From this data, we can determine where coins have come from and where they are going through CiviPort. We can also determine withdrawals that have not been redeemed, as the withdrawal entry will not have a corresponding deposit entry with the same nonce. The nonce is also recorded as a withdrawal on the source bridge and a deposit on the destination bridge. This in turn allows auditors to follow token movements on the respective chains and map them to user addresses. It also allows auditors to query the bridge contract logs to determine timestamps of activity. From a customer service perspective, it allows support personnel to diagnose issues with transfers by being able to see exactly where and when coin movements across chains have been conducted.

KYC/AML

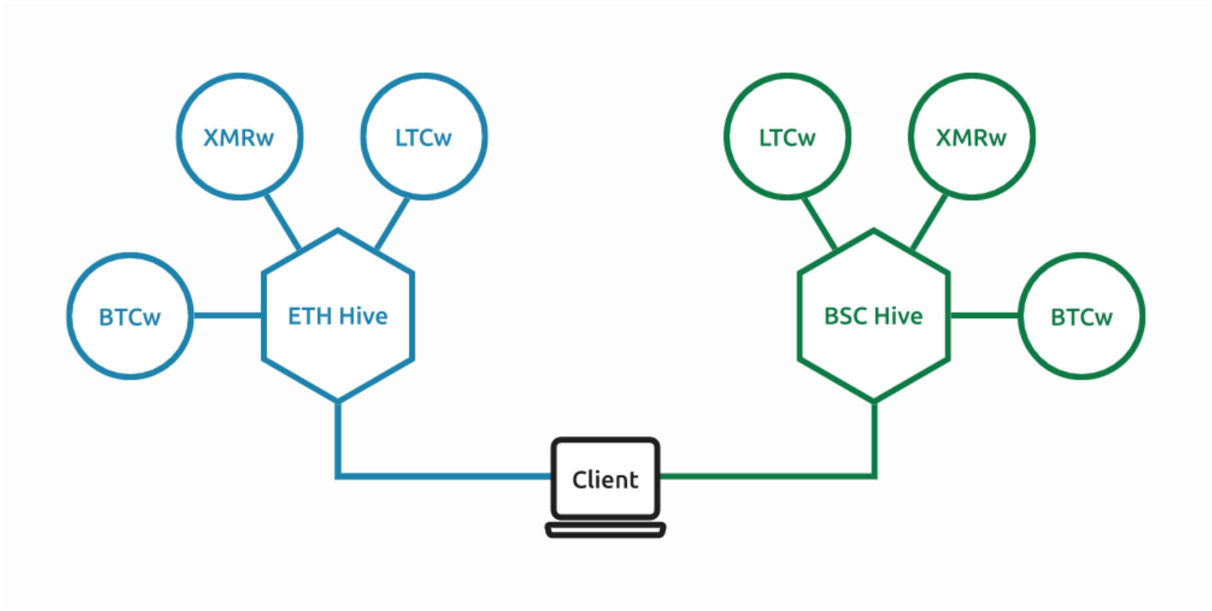
Users will not need to complete any KYC or AML procedures to use CiviPort.

Note about Decentralisation
Currently CiviPort's Ledger and VM are centralized in design but as the community grows the ledger validators will be moved to community appointed representatives chosen through community governance.

CiviSwarm

CiviSwarm is the collective name to the cluster of services that will provide coin wrapping services. CiviSwarm works alongside CiviPort to provide a full stack solution to coin wrapping and cross-chain atomic swaps.

CiviSwarm derives it's name from the structure of the network of services that provide the wrapping across chains. CiviSwarm is a collective or microservice workers configured into hives with each hive running on a physical server. The entry point to the swarm is a hardware server that acts as a gateway, that takes requests to the swarm and routes them to the individual hives, which in turn routes them to the individual workers.



Request Routing

We can send requests to an individual worker in the format

```
https://<hive>.civiswarm.network/<worker>
```

The hive is just a logical collection of workers that exist on the same physical server and provide wrapping services to a particular chain.

Each worker in the hive consists of a smart contract for the wrapped asset and each native asset utilises a shared wallet across the network. A shared wallet allows us to transfer the wrapped tokens across chains with CiviPort and then unwrap and withdraw the native tokens on a different chain without having to concern ourselves with the transport of the underlying asset. Every wrapped token is supported by CiviPort and new wrapped tokens can be added to the CiviPort system without disruption to the network.

This topology also provides layers of redundancy to the network. If a physical hive server were to experience down time it only affects that hive and the rest of the network will continue operation. If an individual worker service were to go offline, then the hive at large is still unaffected. So by breaking down the network into smaller microservices it becomes easier for system administrators to diagnose and correct issues with a particular service and provide a greater level of network security and uptime than would be possible with a monolithic service servicing all chains.

From a software development standpoint this also provides a more scalable solution to supporting further expansion of the network. We don't have to concern ourselves with network or coin selection and monitoring logic across multiple chains. One worker provides wrapping of a single coin for a single network.

The software is built to run as a plugin based system. The native chain integration as well as the destination token chain integration are both plugins to a common software base. This allows us to develop new wrapping capabilities and new destination chain integrations without the need to update the entire network and provides greater scalability and reduced downtime when updating the network.

Native chain integrations will be targeted to primary codebases, with each being able to be configured to support compatible alt coin assets. For example, the btc integration will provide the ability to wrap btc, but then with some additional configuration via a settings file or service parameters, we can support any alt coin which derives its code from btc, such as ltc. xmr for example would allow us to then integrate the dozens of xmr compatible alt coins that exist. This design method allows us to exponentially expand the network with only creating a few core plugins, which can be configured to suit the target network, which will result in a parabolic expansion of the network.

Overall, the combination of CiviPort and CiviSwarm provides a highly configurable, scalable and robust network for native chain asset wrapping and cross chain atomic swaps. This design allows us to exponentially expand the network of native assets and the EVM chains we support by utilizing a few audited building blocks and configuring them to specific networks.

NOTE: The naming convention for Civitas Fundamenta’s wrapped assets will be the Ticker followed by a lowercase w. Example: XMRw, BTCw, LTCw

KYC/AML

As mentioned in the overview to remain compliant with regulations users who wish to wrap/unwrap more than $1,000.00 in value more than once must complete KYC/AML procedures.

Asset Tracking Tokens (ATT)

Asset Tracking Tokens or ATT's will be Civitas Fundamenta's first product to utilize our own wrapped assets. Users will be able to deposit a predetermined set of wrapped assets and mint a Single tradable token. Because a single ATT is representative of ownership of the underlying assets used to create it a net asset value or NAV can be derived. Depending on the price ATT's are trading on DEX's and CEX's the ATT could be either trading at a premium or a discount compared to its underlying assets. This can create arbitrage opportunities between the ATT itself and its underlying assets. When this happens users will have incentive to either create or redeem ATT's to capitalize on the arbitrage in search of profit. This will ideally stimulate trading as well and creation and redemption of ATT's therefore stimulating fees which will be shared with FMTA stakers.

Fees will be paid in the underlying assets when creating and redeeming ATT's and will based on a percentage. The fee level at launch will be 0.3% for creation and redemption of ATT's. This fee will eventually be community governed by CiviDAO.

A more technical document will be released in the near future completely detailing Asset Tracking Tokens.

Rewards and token emissions

Holders of Fundamenta currently have 2 ways to make their tokens work for them. Currently Civitas Fundamenta offers token Staking as well as Liquidity Mining.

Staking

Staking has a maximum cap of 30,000 FMTA staked per account and there is non minimum required to participate. The current return is 73% APY and a saturated account staking 30,000 FMTA will yield 60 FMTA per day. Rewards are accrued approximately every 24 hours (6500 Blocks) and are available 24 Hours after placing your staking position. **Users must have their tokens staked for 48 HOURS** before they are allowed to remove their position. This 48 Hour unlock period is also **RESET** every time you withdraw accrued rewards. If you as a user need unfettered access to your capital **DO NOT** withdraw accrued rewards. Accrued rewards are automatically withdrawn whenever you remove any amount of stake from your position. If users let rewards accrue after 48 hours users can withdraw stake as well as the accrued rewards at any time.

Liquidity Mining

Liquidity Mining has been designed to emit the supply allocated to it rather quickly but this depends on users actually liquidity mining. To date FMTA liquidity Mining has not seen a lot of use so emission from liquidity mining has been relatively small. 10 Million FMTA has been earmarked for liquidity mining and when that number is reached liquidity mining will be tapered or possibly ended altogether.

Users will have 3 lock periods (5,10 & 15 Days) to choose from when liquidity mining all with different return rates. Different lock periods come with different basis point modifiers which are used to calculate liquidity mining rewards. For example the current return using a 15 day lock is 372% for the ETH-FMTA pool and 299% for the USDC-FMTA pool. Returns are slightly better on the ETH-FMTA pool to account for higher levels of risk for impermanent loss.

For Example:

40000 FMTA Added to liquidity on Uniswap with the required ETH to the ETH-FMTA pool would yield 911.23 LP Tokens. Via liquidity mining this would yield:

- 6150.81 FMTA /15 day lock = 410.05 FMTA or 1.02% Daily
- 3462.67 FMTA /10 Day lock = 346.26 FMTA or 0.87% Daily
- 1457.96 FMTA /5 Day Lock = 291.59 FMTA or 0.73% Daily

Liquidity pool tokens are calculated by the following equation:

(amount of FMTA added to Liquidity Pool) / (amount of FMTA in the Liquidity Pool) * (amount of total supply Liquidity Pool (UNI-V2) tokens).

Below are current rates:

USDC-FMTA	**Daily Return %**	**APY %**
5 Day Lock	0.41%	150%
10 Day Lock	0.66%	241%
15 Day Lock	0.82%	299%

ETH-FMTA	**Daily Return %**	**APY %**
5 Day Lock	0.73%	266%
10 Day Lock	0.87%	318%
15 Day Lock	1.01%	372%

Fee Sharing

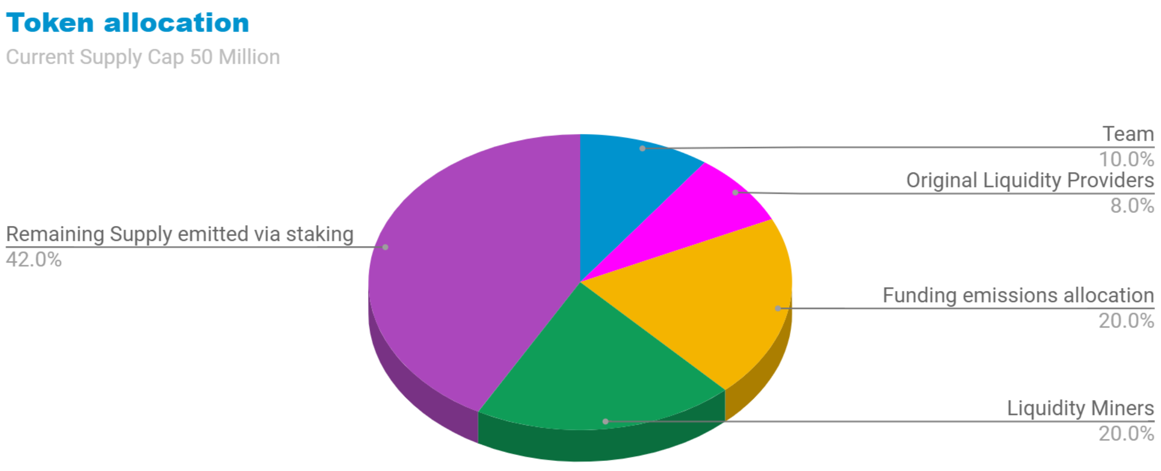
Civitas Fundamenta will be sharing fees from all its ventures with FMTA Stakers. Unlike staking though there will be a to be decided minimum amount required staked so users can take advantage of the fee sharing program.

The Token and it’s tokenomics

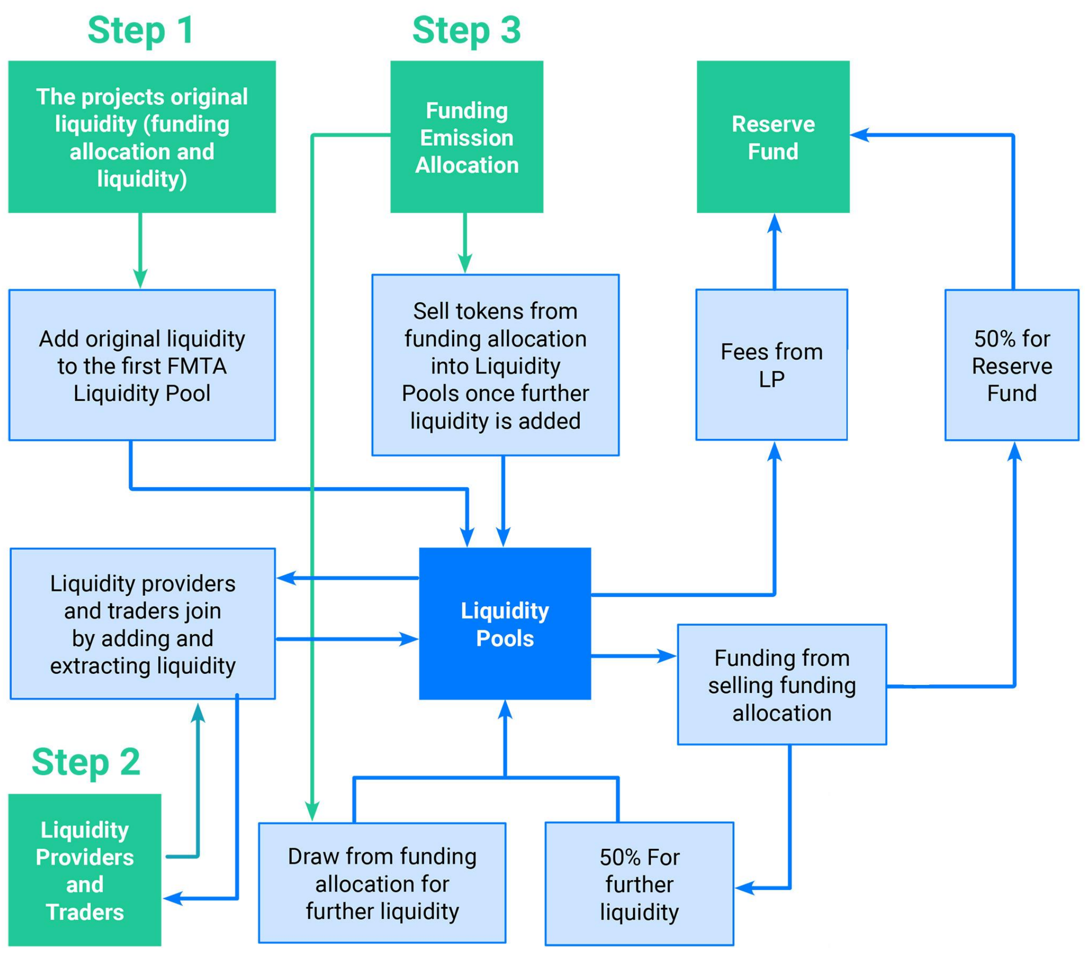
Civitas Fundamenta did not have any presale or private sale. The team and some people we know all came together to start the first liquidity pool on Uniswap. The Fundamenta token was built to be extensible and uses role based access control (RBAC) to securely allow access by other contracts or authorized accounts. The token has mint, burn, mint to and burn from functions which all require RBAC to access. This will allow the ability to adapt and do anything that would require these functions. Fundamenta’s supply cap is also configurable and this makes us unique. In fact, our supply cap at launch back in November 2020 was 100 Million Tokens and it has already been reduced to 50M following community feedback. Eventually the supply cap will be community governed by our DAO in development CiviDAO.

Because there was no presale we are employing a different approach to raise project funding we are calling funding emissions. In times volume and market price are seeing positive movement on DEX’s FMTA from an allocation of tokens are sold. 50% of the funds acquired from the sale are sent to Civitas Fundamenta and 50% are added back to liquidity with the required FMTA to the pool on Uniswap essentially locking in 50% of the value that was removed while emitting coins into circulation by adding back further liquidity. This allows Civitas Fundamenta to raise required funding for marketing, development and general expenses that come with running a business.

Below is the breakdown of the token supply:



Below is a flowchart detailing how funding emissions works.



Step 1
The projects original liquidity (funding allocation and liquidity)
Add original liquidity to the first FMTA Liquidity Pool
Step 3
Funding Emission Allocation
Sell tokens from funding allocation into Liquidity Pools once further liquidity is added
Reserve Fund
Fees from LP
50% for Reserve Fund
Liquidity providers and traders join by adding and extracting liquidity
Liquidity Pools
Funding from selling funding allocation
Step 2
Liquidity Providers and Traders
Draw from funding allocation for further liquidity
50% For further liquidity

10M Tokens allocated to Funding Emissions
10M Tokens allocated to Liquidity Mining
5M tokens allocated to the Team
4M tokens allocated to original liquidity provider of the first pool

This leaves 21 Million tokens to be emitted through Staking.

The Team and original LP allocations had been locked for 6 months after launch and are slated to unlock Fri Apr 16 2021 12:00:00 GMT+0000 but the team and providers have all agreed to a modified unlock schedule. The Developer Team members will receive 60K FMTA Each at the unlock time and then 10K FMTA per month thereafter until they receive their portion in full. The providers and rest of the consultants receive 30K FMTA at unlock time and 5K FMTA per month until their portion is received in full.

This will be a total emission of 390K FMTA at unlock time and total emission of 80K FMTA per month thereafter.

Roadmap

The following are items on our roadmap with no set completion dates as of yet.

CiviDAO

CiviDAO will be our eventual realization of community governance. Civitas Fundamenta wants to be a company that is beholden to its users. Token holders will be able to take part in votes on important company issues as well as everything related to all our offerings. Things like staking rewards, fee levels, our configurable supply cap will all be community governed through CiviDAO.

Before community governance can be implemented, said community needs to grow to a size that is capable of properly governing itself. Because of this there is no timeline as to when CiviDAO will be complete or implemented but the goal is to have this in place as soon as it becomes viable to do so.

CiviCMD

CiviCMD will be a unified interface for users to interact with all of their assets across chains. It will incorporate CiviPort and CiviSwarm providing a single place for users to take advantage of all of Civitas Fundamenta's offerings. Our goal is to ultimately provide a single platform for users to interact with and use their assets in the wider cross chain decentralised finance ecosystem.

CiviLend

If our services see use we will begin to offer over-collateralized loans. This is a distant roadmap item but when completed fees from loans will be shared among FMTA stakers. Because funds are with a custodian we will be able to program access to a portion and offer them in the CiviLend Platform. Because the loans are collateralized the wrapped assets are always backed 1:1 still even with the assets in use. This gives us an advantage over platforms offering cross chain liquidity who do not have access to assets when being used in DeFi.

CiviGrant

Another roadmap item is CiviGrant, a program we will fund with a portion of revenue made from our assets. Through CiviDAO users will be able to choose the amount of revenue allocated to the CiviGrant platform treasury. FMTA Holders will be able to vote on grant allocations from the CiviGrant treasury and then prospective grant recipients will be able to apply with project proposals. FMTA holders will then be able to vote on proposals choosing a winning recipient.

CiviEye

CiviEye is the working name for a set of tools that will be developed to allow auditors as well as users to inspect the assets under Civitas Fundamenta’s management. Our number one priority is and always will be transparency. These tools will allow users to view how many assets are under control and where they are currently being used. For example if Bitcoin is being used by the CiviLend platform, CiviEye will be able to show users and auditors where that Bitcoin is being used, the term of the loan and other pertinent information. While transparency is important, so is user privacy so no user information will be publicly exposed by CiviEye.

Final Thoughts

We know that regulations, compliance, KYC and AML are all dirty words to a lot of users in the world of decentralized finance. That being said the number of people involved in DeFi is actually very small compared to the rest of the wider financial ecosystem as a whole. There is a whole segment of people looking for an insured solution to be involved in DeFi and we want to make it as easy as possible to access while remaining compliant. We also realize that while we are offering a custodial insured option at this time, Decentralisation is extremely important so we will also be working towards a fully decentralized solution not only for wrapped assets, but for CiviPort as well. It's why we will be chain agnostic and support any platform that is viable. We believe we can bridge the divide between compliance and decentralization. Will you join us?