Technical Whitepaper

(Draft v0.2 – For Review)

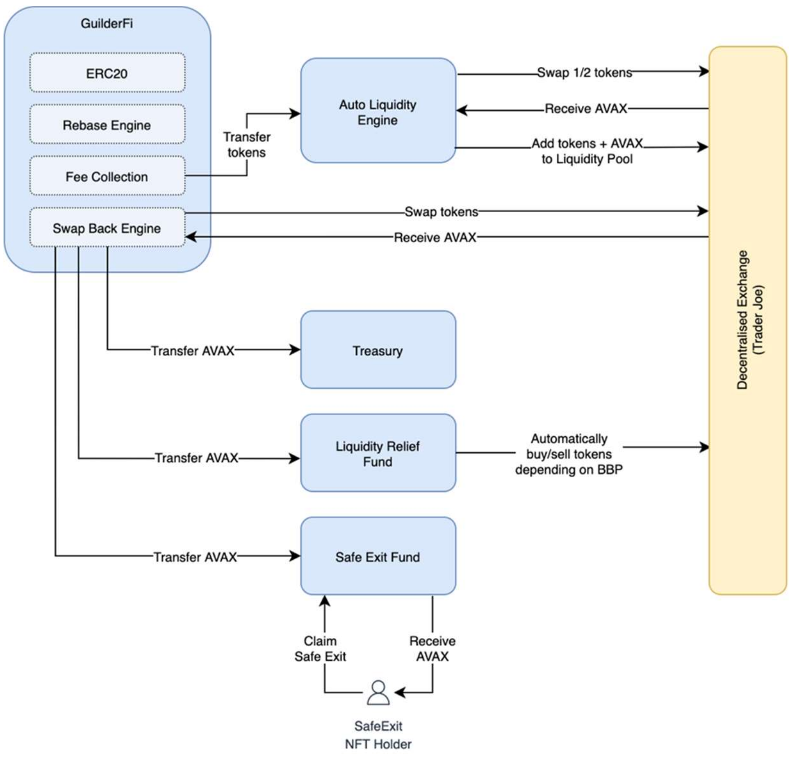
GuilderFi

Buy, Hodl, & Earn

This white paper describes the overall technical solution and application architecture of the GuilderFi protocol.

The main components

The GuilderFi protocol is made up of a number of smart contracts. The main smart contract is the GuilderFi smart contract which implements the ERC20 protocol and integrates with the other contracts.

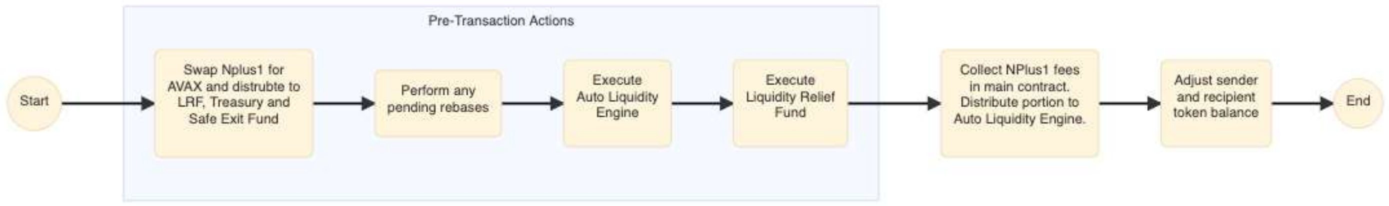


Component	Description
GuilderFi Smart Contract	The main GuilderFi smart contract. This is the ERC20 smart contract that allows users to hold and transact with N1 tokens. A number of features are also implemented in this smart contract such as: • The rebase engine which rewards N1 token holders with additional tokens as per the GuilderFi APY schedule. (See Rebase section for more details) • A fee collection feature which collects fees on every N1 transaction (see Fees section for more details) • A swap back engine which swaps N1 transaction fees with AVAX to be distributed to other components within the GuilderFi protocol.
Auto Liquidity Engine	The Auto Liquidity Engine (ALE) is an additional smart contract that operates within the GuilderFi protocol. A percentage of each N1 transaction is collected as fees through the main contract and are sent to this ALE smart contract. At certain intervals, the ALE smart contract will automatically swap half of these N1 tokens for AVAX and automatically add more AVAX/N1 trading pairs into the Decentralised Exchange (DEX) liquidity pool. The intent is to stabilise liquidity of N1 tokens in the DEX.

Liquidity Relief Fund	The Liquidity Relief Fund (LRF) is an additional smart contract that operates within the GuilderFi protocol. The LRF accumulates AVAX as fees are collected and uses this AVAX plus its N1 token supply to buy and sell tokens through the DEX. See Liquidity Relief Fund section for more details.
Treasury	The treasury functions as additional financial support for the GuilderFi protocol. It helps to establish a floor value for the N1 token and is used together with the LRF to back the token's liquidity giving each token real intrinsic value. The treasury may also be used to fund new products, services, and projects that will expand and provide more value to the GuilderFi community as well as providing funding for marketing. The longer-term goal is to convert the treasury to a fully automated on-chain DAO controlled by N1 holders.
Safe Exit Fund	The Safe Exit Fund (SEF) is an additional smart contract that operates within the GuilderFi protocol. The SFE accumulates AVAX as fees are collected and this AVAX is held within the SFE. Initial token buyers are granted a GuilderFi SafeExit NFT which grants them the ability to safely exit from the GuilderFi protocol and in turn receive back their initial investment of AVAX + a premium. (See Safe Exit section for more details).
Decentralised Exchange (Trader Joe)	The Decentralised Exchange (DEX) plays an important role in the GuilderFi ecosystem. As the GuilderFi protocol will be deployed onto the AVAX network, the DEX that will be integrated into the protocol will be Trader Joe.

Anatomy of a transaction

Every time a transfer of N1 tokens takes place, the following occurs:


Pre-Transaction Actions
Start
Swap Nplus1 for AVAX and distrubte to LRF, Treasury and Safe Exit Fund
Perform any pending rebases
Execute Auto Liquidity Engine
Execute Liquidity Relief Fund
Collect NPlus1 fees in main contract. Distribute portion to Auto Liquidity Engine.
Adjust sender and recipient token balance
End

In a stock standard ERC20 smart contract, the "transfer" function at a minimum should adjust the sender and receiver's token balance within the smart contract (i.e. decrease tokens in the sender's balance and increase tokens in the recipient's balance).

In the GuidlerFi main contract, this is the last step in the transfer flow. A set of "pre-transaction" actions occur before the transfer function reaches this step.

Swapping

As tokens are transferred, the main GuilderFi smart contract will collect tokens and store these within the smart contract. At the start of each transaction, the contract will swap the current balance of N1 tokens (hold within the GuilderFi main contract) with the DEX and receive an amount of AVAX. The AVAX received will be distributed to the LRF, Treasury and Safe Exit Fund.

Note: the tokens swapped in the transaction will not include the tokens being sent as part of that particular transaction. It will only swap tokens that have already been collected in the smart contract from previous transactions. Therefore, the very first transaction will not swap any tokens.

Example scenario:

<u>Transaction 1</u>: Wallet A buys 1000 N1 tokens from the DEX

- Assuming this is the first ever GuilderFi buy/sell transaction, the smart contract would currently have zero N1 tokens accumulated – thus no swapping would occur. Then;
  - 130 N1 tokens are put aside
  - 50 N1 tokens (5%) are transferred to the Auto Liquidity Engine contract address

        - 80 N1 tokens (8%) are transferred to the GuilderFi main contract address. This is based on:
            - 5% LRF Fee
            - 3% Treasury Fee
- The N1 token balance held in the DEX liquidity pair decreases by 1000 N1 tokens.
- Wallet A's N1 balance increases by 870 N1 tokens (i.e. original amount minus 13% fees).

Transaction 2: Wallet B sells 100 N1 tokens to the DEX

- From Transaction 1, the GuilderFi main contract currently holds 80 N1 tokens
- The 80 N1 tokens are swapped on the DEX:
    - 80 N1 tokens are transferred from the Guilder main contract to the DEX liquidity pair (N1/AVAX)
    - A certain amount of AVAX is transferred from the DEX liquidity pair back to the GuilderFi main contract address. The amount will depend on the current price of N1. For simplicity, assume 80 AVAX is returned.
- 50 AVAX is automatically sent to the LRF smart contract address.
- 30 AVAX is automatically sent to the Treasury address.
- The swap is now complete and the rest of the transaction can take place;
    - Sell fees for Transaction 2 are collected and are distributed as follows:
        - 18 N1 tokens are put aside
        - 6 N1 tokens are transferred to the Auto Liquidity Engine contract address (6% fee)
        - 12 N1 tokens are transferred to the Guilder main contract address. This includes:
            - 7% LRF Fee
            - 4% Treasury Fee
            - 1% Safe Exit Fee
    - Wallet balances are adjusted accordingly:
        - Wallet B's N1 balance decreases by 100
        - The DEX liquidity pair balance of N1 increases by 82 (i.e. original amount minus 18% fees).

Note: swapping can be configured to occur on either every transaction or only after a certain period of time has passed, e.g. it can be configured to swap one per day, once per hour, etc.

Rebasing

The GuilderFi smart contract will perform compound rebases every 12 minutes.

How rebase rewards are calculated and distributed

Rebasing is based on the concept of "reflections", whereby the token balance of token holders is adjusted using mathematical formulas rather than looping through each account balance and adjusting the value of each balance.

When the GuilderFi contract is initially deployed, the total supply (100 million N1 tokens) is initially transferred to the treasury. Within the smart contract, the treasury balance is not actually recorded as:

Treasury Balance = 100,000,000

but rather;

Treasury Balance = c, where c is a very large constant number

To explain rebasing, assume that c =10^{18} (or 1 with 18 zeroes, i.e. 1000,000,000,000,000,000). (In reality, the smart contract uses the largest possible unsigned integer that can be represented as 256 bits).

The treasury balance is actually recorded in the smart contract as:

Treasury Balance = 1^{18} or 1,000,000,000,000,000,000

When the balance of an address is requested through the ERC20 "balanceOf" protocol function in the GuilderFi smart contract, the balance is returned by dividing the held balance with a certain number referred to as "gonsPerFragment". This will give the true account balance.

The “gonsPerFragment” variable is calculated as = c / total supply.

Therefore the initial gonsPerFragment value = 10^{18} / 100,000,000 = 10,000,000,000

When the N1 account balance for an address is requested through the GuilderFi smart contract, the contract will dynamically calculate the balance on the fly using the formula:

Actual Account Balance = Account Balance held in the smart contract / gonsPerFragment

Therefore, actual treasury balance = treasury balance held in smart contract / gonsPerFragment
= 1,000,000,000,000,000,000 / 10,000,000,000
= 100,000,000

When a rebase occurs, the total supply of tokens is increased by the compound interest rate. The “gonsPerFragment” value is also re-calculated by dividing c (10^{18}) by the total supply. To understand this in detail, consider the below scenario.

Example scenario:

Initial distribution:

- GuilderFi token is deployed for the first time.
- Total supply = 100,000,000 tokens
- “gonsPerFragment” = c / total supply
  = 10^{18} / 100,000,000
  = 10,000,000,000
- Treasury transfers 10,000,000 N1 tokens to Wallet A
- Treasury transfers 10,000,000 N1 tokens to Wallet B
- Treasury has 80,000,000 N1 tokens remaining

	Balance recorded in smart contract after transfers	Balance returned through ERC20 “balanceOf” function
Wallet A	= 10,000,000 * gonsPerFragment = 100,000,000,000,000,000	= balance in smart contract / gonsPerFragment = 100,000,000,000,000,000 / 10,000,000,000 = 10,000,000
Wallet A	= 10,000,000 * gonsPerFragment = 100,000,000,000,000,000	= balance in smart contract / gonsPerFragment = 100,000,000,000,000,000 / 10,000,000,000 = 10,000,000
Treasury	= 80,000,000 * gonsPerFragment = 800,000,000,000,000,000	= balance in smart contract / gonsPerFragment = 800,000,000,000,000,000 / 10,000,000,000 = 80,000,000

After first rebase:

- For illustration purposes, assume rebase rate = 0.016%
- The total supply is increased by the rebase amount, i.e.
  - Total Supply = Total Supply * $(1 + 0.016\%)^1$
    = 100,000,000 * $(1 + 0.016\%)^1$
    = 100,016,000
- “gonPerFragment” is re-calculated by calculating:
  - “gonsPerFragment” = c / Total Supply
    = 10^{18} / 100,016,000
    = 9,998,400,256

	Balance in smart contract before rebase	Balance returned through ERC20 “balanceOf” function after rebase
Wallet A	100,000,000,000,000,000	= balance in smart contract / gonsPerFragment = 100,000,000,000,000,000 / 9,998,400,256

		= 10,001,600
Wallet B	100,000,000,000,000,000	= balance in smart contract / gonsPerFragment = 100,000,000,000,000,000 / 9,998,400,256 = 10,001,600
Treasury	800,000,000,000,000,000	= balance in smart contract / gonsPerFragment = 800,000,000,000,000,000 / 9,998,400,256 = 80,012,800

When a rebase occurs, rather than the smart contract looping through each wallet address and increasing each balance by 0.016%, the contract will adjust the total supply and "gonsPerFragment" variable. Each account balance is therefore adjusted accordingly when the balance is next requested.

Manual rebasing vs auto rebasing

When smart contracts update their state they require a "write" transaction to be executed on the underlying blockchain which requires gas fees. Smart contracts do not have the capability to perform scheduled transactions and this requires either a human or off-chain software to perform a blockchain transaction at scheduled intervals.

Whenever an N1 transfer occurs, the smart contract will check when the last rebase was calculated. If 12 minutes or more have passed since the last rebase, the smart contract will attempt to automatically perform a rebase.

Example scenario #1

- Last rebase occurred at 9:00am
- The next transfer occurs at 9:05am
    - When this transfer occurs, the smart contract sees that the last transfer was < 12 minutes ago
    - No rebase is automatically performed

Example scenario #2

- Last rebase occurred at 9:00am
- The next transfer occurs at 9:13am
    - When this transfer occurs, the smart contract sees the last transfer was >= 12 minutes ago
    - The smart contract executes a rebase
        - The total supply is automatically increased by $(1 + 16\%)^1$

Example scenario #3

- Last rebase occurred at 9:00am
- The next transfer occurs at 10:01am
    - When this transfer occurs, the smart contract sees the last transfer was >= 12 minutes ago
    - The smart contract calculates that 5 rebases should have occurred in the last hour
    - The smart contract executes 5 rebases
        - The total supply is automatically increased by $(1 + 16\%)^5$

If there are enough regular transactions, rebasing should automatically occur after the 12 minute interval is reached and the next transaction is executed.

However, if there are not enough transactions occurring, the number of pending rebases will start to accumulate over time. If there are too many rebases that are required, the smart contract will attempt to perform a maximum of 40 rebases. This is to ensure the gas fees to execute a transfer do not become so large that the token is no longer able to be transferred.

Example of rebase capping:

- Last rebase occurred at 9:00am
- The next transfer occurs at 9:01pm (12 hours later)

    - When this transfer occurs, the smart contract sees that 60 rebases should have occurred since the last rebase
    - The smart contract automatically executes 40 rebases
        - The total supply is automatically increased by $(1 + 16\%)^{40}$
- Another transfer occurs at 9:02pm
    - When this transfer occurs, the smart contract sees that there are still 20 rebases that should have occurred since the last rebase
    - The smart contract automatically executes the remaining 20 rebases
        - The total supply is automatically increased by $(1 + 16\%)^{20}$

As a safety precaution, the GuilderFi team will run a scheduled off-chain scheduled tasks to run every 8 hours. The intention of this scheduled task is to "nudge" the smart contract to perform any pending rebases in case there are not enough transactions to automatically trigger rebases.

Note: the rebase function in the GuilderFi smart contract is a public function and can be triggered by any user who is willing to pay the gas fees to execute it. If there are no pending rebases (i.e. the last rebase was <12 minutes ago), the function will not require any gas fees.

Liquidity Relief Fund

The Liquidity Relief Fund (LRF) is a separate smart contract that will accumulate AVAX as N1 fees are collected and swapped for AVAX. The LRF can be configured to executed along with every N1 transaction or after a certain period of time has passed, e.g. execute the LRF once a day, once an hour, etc.

When the LRF is executed, it will first calculate the backed liquidity ratio. This ratio is calculated as:

Backed liquidity ratio = (AVAX balance of treasury + AVAX balance of LRF) / Amount of AVAX in the liquidity pool

Example:

- AVAX in treasury = 50 AVAX
- AVAX in LRF = 50 AVAX
- AVAX in liquidity pool = 100 AVAX
- Backed liquidity ratio = 100%

Note: the LRF will also will be transferred a fixed amount of N1 tokens when the GuilderFi contract is first deployed.

The LRF's role is to buy and sell N1 tokens and ensure the backed liquidity ratio remains at 100%.

Example 1: < 100% Backed Liquidity:

- To start with:
    - AVAX in treasury = 50 AVAX
    - AVAX in LRF = 50 AVAX
    - AVAX in liquidity pool = 120 AVAX
    - Therefore, backed liquidity ratio = 100 / 120 = 83.33%
- On the next transaction, the LRF calculates that it needs to sell 10 AVAX worth of N1 tokens to bring the ratio to 100%
    - LRF sells 10 AVAX worth no N1 tokens
    - DEX liquidity pair decreases its AVAX supply by 10 AVAX
    - LRF increases its AVAX supply by 10 AVAX
- After the LRF executes:
    - AVAX in treasury = 50 AVAX
    - AVAX in LRF = 60 AVAX
    - AVAX in liquidity pool = 110 AVAX
    - Therefore, backed liquidity ratio = 110 / 110 = 100%

Example 2: > 100% Backed Liquidity:

- To start with:

    - AVAX in treasury = 50 AVAX
    - AVAX in LRF = 50 AVAX
    - AVAX in liquidity pool = 80 AVAX
    - Therefore, backed liquidity ratio = 100 / 80 = 125%
- On the next transaction, the LRF calculates that it needs to buy 10 AVAX worth of N1 tokens to bring the ratio to 100%
    - LRF buys 10 AVAX worth no N1 tokens
    - DEX liquidity pair increases its AVAX supply by 10 AVAX
    - LRF decreases its AVAX supply by 10 AVAX
- After the LRF executes:
    - AVAX in treasury = 50 AVAX
    - AVAX in LRF = 40 AVAX
    - AVAX in liquidity pool = 90 AVAX
    - Therefore, backed liquidity ratio = 90 / 90= 100%

Note: the LRF smart contract will only start executing these buy/sells when the backed liquidity ratio reaches >=100% for the first time.

Also note: the 100% ratio can be triggered by either:

- The AVAX in the LRF and treasury growing over time until it reaches the amount of AVAX in the liquidity pool
- Or the AVAX in the liquidity pool decreasing over time if there are more sells than buys
- Or a combination of both scenarios

Once the 100% ratio has been triggered, the LRF will continue to buy/sell as long as it falls within a configurable range. This range will be initially set to 85% - 115%, meaning that if the backed liquidity percentage falls between this range, the LRF will automatically buy and sell tokens. If the backed liquidity is lower or higher than this range, the smart contract will take no action. In this case, the treasury account has access provisioned to either withdraw or buy/sell tokens in the LRF smart contract as needed manually to bring the percentage back within acceptable range.

Fees

Fees are collected whenever N1 tokens are bought or sold from the DEX. Fees are not applied to transfers between wallet addresses.

The GuilderFi smart contract caters for the 5 following types of fees:

1. LRF fees
2. Treasury fees
3. Auto liquidity engine fees
4. Safe exit fees
5. Burn fees

The GuilderFi contract allows the treasury account to modify the fees individually for buys and sells.

Note: At the moment, burn fees are set to zero but can be updated to automatically burn a portion of N1 tokens with each transaction if needed.

The smart contract has a hard-coded cap of:

- 20% for buy fees
- 25% for sell fees

The treasury account has provisioned access to update the fee percentages buy/sell fees for the 5 fee categories as long as the total is <= the hard-coded cap.

Safe Exit Fund

The Safe Exit Fund is a unique feature of the GuilderFi protocal and the end to end technical design is currently in the process of being finalised.

At a high level the basic concept of Safe Exit is to reward all pre-sale token buyers with a safety net that allows them to recoup their initial investment.

When the GuilderFi smart contract is first deployed, trading will be restricted, meaning only whitelisted addresses may transfer N1 tokens. The only address whitelisted to transfer tokens at the very beginning will be the treasury address.

During the pre-sale, trading will continue to be “closed”. Once the pre-sale has finished and N1+AVAX have been added to the DEX liquidity pool, trading can be manually “opened” by the treasury account. At this point, all current token holders should be able to claim a Safe Exit NFT.

This NFT will allow holders to trade their Safe Exit NFT to recoup their initial AVAX investment + a 6.25% premium.

Example scenario:

- Wallet A purchases 1000 N1 for 100 AVAX in pre-sale
- Wallet A is given a Safe Exit NFT
- The Safe Exit fund accumulates AVAX over time through fees
- Wallet A can use their NFT to receive 106.25 AVAX (assuming the Safe Exit Fund has accumulated enough AVAX to pay the claim)
  - The NFT becomes unusable – no one else is able to use it to claim Safe Exit
  - The total N1 balance of the NFT holder is burnt

Other things to consider:

- If the NFT holder transfers or sells any amount of their N1 token, their Safe Exit NFT becomes invalid and cannot be used to claim Safe Exit.
- A Safe Exit NFT holder may decide to trade their NFT with another person. The person who receive the Safe Exit NFT will be able to use the token to recoup any “buys” from that point forward.

Please note: this documentation will be expanded further as the Safe Exit feature is finalised.