DeFi Chain Modules

Introduction

There are two criteria to create value for a public chain. Assets Under Management (AUM) and feature set. A feature set is simply a collection of functions that can be acted upon the assets. A chain that is feature rich, but has no assets will ultimately not create value. Inversely a chain with high AUM but lacking features will fail to capture NVT.

The following document aims to align these two criteria and stand as a proposal to enhance an IBC sidechain or direct module integration, the features offered by the chain can yield high value to the DeFi space.

Feature Set

The following is a high level description of the base instruments made available via the technological modules that have been designed. Following this section there will be examples of how these instruments can be used to create DeFi and decentralized derivative markets.

- Auctions
- Loans
- Collateralized Tokens
- Escrow (Deposit, Future, Lock based)
- Token Issuance
- Interest bearing token issuance
- Non Fungible Token collection creation
- Proof of Existence
- Protocol level automic swaps

The above are the raw instruments required to build a variety of traditional finance tooling on top of the existing DeFi ecosystem.

Traditional Use Cases

Coins for Open Finance

- Increases the Bank's FIAT Deposits (Liquidity) by attracting customer funds
    - Offer customers' new utility on funds (Transacting)
    - Provides new savings products (Fixed Term / Call Accounts)
    - Application for Securitisation (Collateralisation)
- Readiness for Central Bank Digital currencies
    - Commercial Bank Stable Tokens
    - Commercial Bank money as well as Coin and Note on and off-ramp to CBDC

Tokenised Markets for Investment Banks

- Primary Debt & Equity Markets
    - Issuance, Book Building, Auction, Interest Payment, and Term Renegotiations
- Secondary Debt & Equity Spot Markets
- Alternative Token Markets
    - Gold, Silver, Platinum, Diamonds, Oil, and more
- Tokenised P2P Synthetic Futures
    - No exportation of value, capital flight, or BoP requirements
    - Forex Futures (Hedging / Cover) Markets
    - Global Shares and/or Commodity Futures Markets
    - Carbon Credit / Green Energy Futures Markets
    - Equity Future Markets

Digital Currency for Central Banks

- Net Interbank Settlements
- Treasury Notes, Bills and Bonds Management
- Foreign Stable Token Exchange
    - B2B Cross Border Payments
- Mass Market Digital Currency
    - Remittance of Social Grants

Managing Debentures Issuing

- Debt tokenisation
- Debt modelling through capital and interest tokens
- IOI / book building
- Auction / sale processes
- Extension to support Equity Token issuance

DeFi Use Cases

Liquidity based pool swaps https://uniswap.io/

Compound based lending pools https://compound.finance/

Multi asset collateralized debt tokens https://cdp.makerdao.com/

Non-inflationary, stable, multi asset staking

DeFi Staking

Current prevalent problems with staking solutions are;

Inflationary

More tokens are minted as reward for validators providing security to a network. This only works if you measure your rewards in the inflationary token, however server costs are measured in fiat, for the sake of simplicity, USD. When a token's supply inflates, it normally suffers a direct inverse correlation with price. If you double supply, you would halve the price. So the net result for validators is often negative. Cost in USD-USD value of tokens received over time.

Inflationary models are also unsustainable as you can't infinitely keep minting new tokens, supply should be limited to ensure ongoing viability for early adopters.

Stable Rewards

As mentioned in the previous section, server costs are measured in USD, however rewards are in tokens, to offset this, you need to sell tokens, when a tokens price depreciates to the point where it no longer covers the opex expenses, you need to discontinue the service. For this reason, staking is mostly a very mutable solution where staking farms will move from one coin to the next to ensure maximized profit due to fluctuating prices.

The role of tokens as network security

A token provides security for its network. If you have a public blockchain that tokenized real-estate. And let's assume it has $200M worth of tokenized real-estate. However, the underlying token staking value is worth $2M. Then in a staking model, it would cost you ~$2M to attack an object of value equal to $200M. The value of the underlying securitized token is what protects the network and its assets. So the value of the underlying security assets, should exceed, or at least match, the value of the managed assets.

Design

To circumvent this, we designed a system that has;

- Stable Rewards
- Multi asset support
- Non-inflationary

This was achieved via 2 popular DeFi mechanisms, collateralized debt, and liquidity pool lending.

Collateralized Stable Debt Tokens

Collateralized Stable Debt Tokens or CSDT are USD pegged stable tokens minted from provided collateral. This allows validators / delegators to provide collateral in any native supported asset (BTC/ETH/BNB on genesis), and mint CSDT. The collateralized ratio of each asset is set via governance and on genesis would be BTC 150%, ETH 150%, BNB 100%

A practical example of how this works;

We deposit 1,000 BNB into our Collateralized Debt Contract, the validators via their approved oracles have agreed (via consensus) that the current price for 1 BNB is $20. The value of the deposited BNB is $20,000. BNB is collateralized at 100%, so the minting value would be $20,000, or 20,000 CSDT that can be minted.

An important disclaimer here, it is best practice to not mint the maximum allowed as this sets a very high liquidation price for your assets. The way the stability of the price is managed is via the liquidation of assets should the token price depreciate. In the above example, should BNB price fall to $18 the collateral BNB would be liquidated and made available for auction at 5% discount, so that another user could purchase the BNB for CSDT (CSDT goes back into the system, your collateral debt is paid off, and they now own the BNB). If however in the example above the collateral provider instead decided to mint 5,000 CSDT, then the liquidation price would be ~$10 and they have a lot more safety and room to manage their collateral.

Stable Staking

With CSDT created, we can now create a validator by staking CSDT. Rewards will be paid out in CSDT. Current rewards are set at 4% with a staked target of 67%. So if there is 1,00 CSDT minted, and 67 are staked the reward will be 4% annually. Should more than 67% be staked, this will decrease to 2% over time, and should less than 67% be staked, then this will grow to 6% over time.

Since CSDT is USD pegged, it means that your returns are fixed against your operational costs.

Non-Inflationary

Now that there is a liquidity pool available of assets (via the Collateral Debt Contracts) it means that users of the system can borrow assets. This is collateralized lending with the ratio set per supported asset. Additional assets over and above CSDT assets can be supported.

Practically, this would mean a borrower can provide BTC as collateral, and borrow ETH against this, with the interest being set based on the ratio of liquidity providers to borrowers. Judging by current rates, this is around ~4%. This interest is then paid to the liquidity providers (stakers) and offsets that value of staking rewards.

Eventually with network growth and liquidity providers, this will offset the ~7%–10% of required staking rewards and arrive at a non-inflationary model.

Modules

The following is the high level and technical description of the designed modules

Module Auction

Generic module for creating auctions and allowing users to place bids until a timeout is reached

StartForwardAuction starts a normal auction. Known as flap in Maker.
StartReverseAuction starts an auction where sellers compete by offering decreasing prices. Known as flop in Maker.
StartForwardReverseAuction starts an auction where bidders bid up to a maximum bid, then switch to bidding down on price. Known as flip in Maker.
StartAuction triggers the creation of an auction on a lot and subtracts coins from the initiator.
PlaceBid places a bid on any auction

Utility functions include;

CloseAuction
GetNextAucitonID
IncrementNextAuctionID
SetAuction
GetAuction
DeleteAuction
GetNextAuctionIDKey
GetAuctionKey
InsertIntoQueue
RemoveFromQueue
GetQueueIterator

GetAuctionIterator
GetQueueElementKeyPrefix
GetQueueElementKey

Use Cases;

Auctions
CSDT liquidation auctions

Module Authority

Generic module for setting system wide authority / managers for issuers and liquidity providers

SetAuthority allows an existing authority to add another authority
CreateIssuer allows an authority to make an account an issuer for specific denoms. Multiple issuers can exist for a single denom.
DestroyIssuer removes the issuance rights for the account

Utility functions include;

GetAuthority
MustBeAuthority

Use Cases

Management of Native tokens

Module Compound WIP (Module Pool is the current working implementation)

Collateral lending module based on borrowing from liquidity pools and paying liquidity providers with interest.

Users can provide liquidity to liquidity pools. This liquidity also acts as collateral. If the user does not borrow against the collateral they receive interest. Interest is calculated based on liquidity to borrowing ratio. The less liquidity available for borrowers the higher the interest rates. Interest is split between all liquidity providers for a given asset. So borrowing interest rates will be higher than liquidity provider interest received.

DepositFundFromAddress allows the user to deposit any denom they own as liquidity / collateral

Price values are set in CSDT (explained below, for simplicity sake this is a USD pegged stable coin). Each denom has a set collateral ratio. Assuming BNB at 150%, means you have to provide $1.5 worth of BNB to borrow $1 worth of ETH. It is recommended to over collateralize to avoid liquidation.

WithdrawFundToAddress allows the user to take the specified amount from the pool and store into the given account balance as long as they have sufficient collateral.

Interest is accrued every block or 10 seconds, whichever is longer.
DistributeReward distributes the given reward between all the funders

Use cases

Decentralized collateralized lending, required for decentralized **shorting** solutions

Module CSDT

Collateralized Stable Debt Tokens (CSDT) that are asset pegged (asset is USD, but can any other asset, BNB for example).

Similar to compound, but non-interest bearing for liquidity providers (currently, this will be enhanced with interest bearing accounts from Module Compound in the future as these are shared liquidity pools)

Allows the creation of CSDT from multi asset collateralization. Assets supported are set via governance and genesis. Recommended BTC, ETH, BNB, and potentially some stable coins.

Users provide collateral. The collateral has price oracles that submit the current value. Users can then withdraw debt (CSDT) equal to their risk appetite or liquidation. The closer to 100% of the collateral value the closer the liquidation point becomes.

Once collateral is no longer enough to support the debt, the lot goes up for liquidation and eventually auction.

Should the collateral value increase, more debt can be withdrawn against it.

CSDT is the recommended supported staking token for the chain, as it supports multiple assets.

Module supports setting global debt limits as well as per asset debt limits to minimize volatility risk in certain assets

Debt Limits, Collateral assets accepted, and collateralized ratio can all be controlled via governance.

This module is being expanded to add support for NFT objects as well that can be used as collateral, although this functionality is not currently available, ETA ~ 2 weeks.

Utility functions include;

ModifyCSDT

PartialSeizeCSDT
ReduceGlobalDebt
GetParams
SetPArams
GetCSDTKeyPrefix
GetCSDTKey
GetCSDT
SetCSDT
DeleteCSDT
GetCSDTs
GetGlobalDebt
SetGlobalDebt
GetCollateralStateKey
GetCollateralState
SetCollateralState
GetLiquidatorAccountAddress
AddCoins
SubtractCoins
Getcoins
HasCoins
GetLiquidatorModuleAccount
SetLiquidatorModuleAccount
StripGovCoin

Use Cases

Multi asset staking tokens
Collateralized debt (NFT or other)
Can collateralized debt into collateralized debt (CDO)

Module Escrow

Allows for the creation of account based escrow locks. Three types are supported, classic 2 party deposit escrow, future escrow and lock based escrow.

Deposit - two parties need to deposit a prefixed value, and if both parties have matched the deposit funds are swapped.

Future - based on a time based trigger.

Users can create lock boxes to deposit denoms into, these boxes have set conditions, either time, multi party, or oracle conditions to release. The box itself is a tradeable asset and can be used as a derivative of the underlying asset.

Utility functions include;

GetDepositedCoinsAddress
SendDepositedCoin
CancelDepositedCoin
SubDepositedCoin
GetDepositedCoin
SetBox
SetAddress
SetName
AddBox
Fee
GetBox
GetBoxByOwner
GetBoxByAddress
CanTransfer
SearchBox
List
ListAll
Iterator
CreateBox
ProcessInjectBox
ProcessBoxWithdraw
SetBoxDescription
DisableFeature
DisableTransfer
SendCoins
GetIdsByName
GetIdsByAddress
ActivateBoxQueueIterator
InsertActiveBoxQueue
RemoveFromActiveBoxQueue
ProcessLockBoxCreate
ProcessLockBoxByEndBlocker
PRocessFutureBoxCreate
ProcessFutureBoxInject
InjectFutureBox
CancelDepositFromFutureBox
ProcessFutureBoxDistribute

ProcessFutureBoxByEndBlocker
ProcessFutureBoxInjectByEndBlocker
ProcessFutureBoxWithdraw
ProcessFutureBoxActiveByEndBlocker

Use Cases

Escrow
Futures
Forwards
Options

Module Interest

Interest module allows issued coins to accrue interest, interest can be set on a per denom level.

SetInterest allows the owner to set fixed interest rates for tokens.
MintCoins allows the module to generate tokens based on interest for account holders.

Utility functions include;

GetState
SetState
SetInterest
TotalTokenSupply
MintCoins
AddMintedCoins

Use cases

Interest bearing tokens
Interest bearing savings accounts

Module Issue

Allows for the creation of ERC20 standard mapped token issuance with fixed (governance controlled) creation and management fees.

CreateToken allows the creation of a token, the denom itself is not reserved for the symbol, but instead an incrementing ID prefixed, for example bnb128e12c while the metadata preserves the symbol, for example FTM

Mint allows mintable tokens to be minted, only if the token is created as a mintable asset

BurnOwner allows the owner to burn tokens they hold
BurnHolder allows the token owner to burn tokens from another address
Freeze allows the owner to lock transfers from an address (In, Out, or In & Out)
TransferOwnership allows a token owner to transfer ownership to another address
Approve allows a token holder to approve another account to use up to a certain limit of tokens in the account
IncreaseApproval allows the token holder to increase the token limit set in Approve
DecreaseApproval decreases the allowance
SendFrom allows the owner to transfer tokens from another address

Parameters per token are as follows;

Name
Symbol
TotalSupply
Description
BurnOwnerDisabled
BurnHolderDisabled
BurnFromDisabled
MintingFinished
FreezeDisabled

Global parameters (set via governance) are as follows;

CreateFee
MintFee
FreezeFee
BurnFee
BurnFromFee
TransferOwnerFee
DescribeFee

Utility functions include;

SetIssue
SetInterestRate
SetAddressIssues
DeleteAddressIssues
RemoveAddressIssues
AddAddressIssues
SetSymbolIssues
SetFreeze
SetApprove
AddIssue

CreateIssue
Fee
GetIssue
GetIssues
SearchIssues
List
Iterator
ListAll
GetIssueByOwner
FinishMinting
DisableFeature
DisableBurnOwner
DisableBurnHolder
DisableFreeze
DisableBurnFrom
CanMint
Mint
BurnOwner
BurnHolder
Burn
BurnFrom
GetFreeze
GetFreezes
Freeze
UnFreeze
FreezeIn
FreezeOut
FreezeInAndOut
SetIssueDescription
TransferOwnership
Approve
IncreaseApproval
DecreaseApproval
CheckFreeze
SendFrom
Allowance
GetAddressIssues
GetSymbolIssues
SetParams
GetPArams
SetInitialIssueStartingIssueID
GetLastIssueID
GetNewIssueID

PeekCurrentIssueID

Module Issuer

Authority module to control native denomination tokens. Issuers are created via the Authority Module and are assigned specific denoms to control. They can set the credit or decrease the credit allowed by liquidity providers.

IncreaseCreditOfLiquidityProvider increases the credit for a specific denom for a specific address
DecreaseCreditOfLiquidityProvider decreases the credit for a specific denom for a specific address
RevokeLiquidityProvider removes the liquidity provider address
SetInterestRate sets the inflation rate of the denom, if any

This module allows minting / burning of native tokens, such as the genesis tokens created on-chain.

Utility functions include;

GetIssuers
SetIssuers
AddIssuer
RemoveIssuer
ValidateDenoms
CollectDenoms
MustBeIssuer

Module Liquidator

The liquidator module settles bad debt from undercollateralized CSDT's by seizing them and raising funds through auctions.

SeizeAndStartCollateralAuction pulls collateral out of a CSDT and sells it in an auction for CSDT. Excess collateral is goes back to the contract
StartDebtAuction sells off minted gov (native) tokens to raise set amount of stable coin (if collateral needs to be bought to keep the price of CSDT stable)
SettleDebt removes equal amounts of debt from the liquidators reserves

Global parameters allows governance to set max auction debt sizes for removed collateral (to ensure smaller lots for bidding)

Utility functions include;

partialSeizeCSDT

GetParams
SetParams
GetSeizedDebt
SetSeizedDebt

Module Liquidity Provider

Liquidity provider is an internal module used for module accounts to have set credit limits and allow the module to mint tokens against credit that they own. This is used for liquidity pools, uniswap, compound, liquidator, and csdt's as a vendor of last resort. Users can also donate to the liquidity provider to keep the system stable.

CreateLiquidityProvider allows the creation of a new account that acts as a liquidity provider
BurnTokensFromBalance allows the provider to burn tokens it owns
MintTokensFromCredit allows the provider to mint tokens against assigned credit (can be from an external - non-chain source)

Utility functions include;

SetLiquidityProviderAccount
RevokeLiquidityProviderAccount
GetLiquidityProviderAccount

Use cases

Credit providers based on assets held
Stable coins
Asset backed credit

Module NFT

Allows for the creation of Non Fungible Token collections and Non Fungible Tokens (NFTs)

MintNFT creates a new NFT object with a collection and owner(s)

Utility functions include;

DeleteNFT
UpdateNFT
GetNFT
SetCollection
GetCollection
GetCollections

GetDenoms
GetOwners
GetOwner
GetOwnerByDenom
SetOwnerByDenom
SwapOwners

Use cases

Debt Issuance
Debentures

Module Pool WIP for Compound Module

Pool module acts as a global shared interest bearing liquidity pool for any supported system asset.

See Module Compound

Utility functions include;

GetAccountFunds
DistributeReward
GetTotalFunds

Use cases

Loans
Shorts
Hedging

Module Pricefeed / Oracle

Allows a group of white-listed oracles to post price information of specific assets that are tracked by the protocol.

AddOracle allows governance to add additional white-listed oracles
AddAsset adds an asset to track it's price / value
SetPrice allows an Oracle to submit information with regards to the tracking value

After each block all submitted prices are aggregated and a median (weighted by stake) value is created as the current price.

Utility functions include;

SetCurrentPrices sets the current aggregate prices after a block finalizes
GetOracles
GetAssets
GetAsset
GetOracle
GetCurrentPrice
GetRawPrices returns all submitted raw oracle prices
ValidatePostPrice

Module Record (Proof of Existence)

Allows the creation of proof as an immutable record.

CreateRecord creates a record set, for example University Certificates
AddRecord allows the storing of a record with metadata. Such as a unique certificate

Utility functions include;

GetRecord
GetRecordByID
List
GetAddressRecords
GetRecords
GetIteratorRange
SetInitialRecordStartingRecordId
GetLastRecordID
GetNewRecordID
PeekCurrentRecordID

Use cases

Luxury good proofs
Certification proofs
Anti fraud mechanism

Module Uniswap

Generic system uniswap protocol. Users can provide liquidity and set pool based swap ratios.

AddLiquidity allows a user to add liquidity to a pool. User can specify the token they wish to swap for.

Swap allows a user to swap from a liquidity pool for the swap requested asset for the counter pool.

This allows swaps based on liquidity ratio in the pool, so if the pool is 100:1 and you add 100, you can withdraw 1.

Utility functions include;

SwapToCoin
SwapFromCoin
SetPool