AssetMantle Whitepaper

AssetMantle¶

Redefining Digital Asset Ownership

Non-fungible tokens (NFTs) have experienced massive adoption within music, gaming, collectibles, physical arts, and beyond. These NFTs act as a certificate of ownership and thus requires a standardized representation of these tokenized assets.

The majority of the NFT-related applications depend on centrally administered databases with fragmented capture of information. Being centrally operated the applications allow for high TPSs when the operations are internal to the system but become painfully slow when reconciling transactions with other applications. Considerable efforts have to be respent on KCL/AML at each onboarding of the same individual due to lack of interoperability. The lack of transparency into the data of these applications also allows for manipulation and discourages auditability.

AssetMantle solves these problems of traditional finances by bringing auditability and interoperability between applications through decentralization solutions. It makes it easy to define open finance applications through its toolkit:

- Node
- : a set of modules that can be used together to implement a chain'snode application
- .
- Client
- : a set of dApp utilities like transaction queuing and ordering mechanisms, custom transaction explorers, and centralized clients built to help build applications end-to-end.
- Spec
- : a standard specification for defining NFTs, RFTs, FTs, Wallets, etc to allow for a common set of transactions facilitating interoperability between all the applications built on the ecosystem.
- Core
- : a common application that all the other applications in the ecosystem connect to as a raised and open moderator for interoperability, shared security, and as a gateway to other ecosystems as well.
- Integrations
- : a set of integrations baked into the platform to allow for common utilities like KYC verification, and fiat on/off ramps.
- Validators
- : a curated set of reputed validators with a sizable amount of assets under delegation on other DPoS applications that can be utilized to ensure security and unbiased validation and trust on application chains.
- Interoperability transactions
- : a common set of transactions that all applications in the ecosystem have to implement to participate and be interoperable with other applications in the ecosystem.

Table of Contents¶

1. Overview¶

AssetMantle is an interoperability middleware for building a decentralized infrastructure for, but not limited to, the creator economy. The AssetMantle platform is a set of standardized tools and specifications that can be used to model exchanges and marketplaces by tokenizing, trading, and financing against 'real-world' financial assets (such as invoices, receivables purchase agreements, bills of lading/commodities, letters of credit, solar credits, etc.). Since all the exchanges are built using the same set of AssetMantle modules and specs, there is native interoperability between all the exchanges while maintaining sovereignty. The decentralized and open finance features of AssetMantle applications are leveraged for more efficient capital allocation in a trust-minimized, borderless and 24/7 manner.

This new paradigm of novel use-cases can be facilitated within the AssetMantle Ecosystem using its continuously evolving toolset, which currently includes modules like reputation, contract, escrow, access control, hierarchy, NFT, and RFT.

Features of the AssetMantle ecosystem:

- Native interoperability between all applications participating sovereignly in the ecosystem
- The applications are also interoperable across different protocols/platforms
- Singular representation of data, i.e. each object is allowed to exist only once in the whole ecosystem
- Flexibility in the level of transaction validation privacy level, covering the whole spectrum from open public validation to permissioned private validation
- Privacy-preserving transaction validation allows interoperability between publicly and privately validated dApps
- Interchain NFT Transfers
- Multi-Wallet Support/Wallet Agnosticism
- Customize NFT Metadata
- Reduced Load on Interchain Protocol
- Privacy Protection

1.1. Design Goals¶

- Interoperability
- : dApps built using AssetMantle should have native interoperability enabled as an option.
- Flexibility
- : The spec definition should be flexible enough for a range of applications to be built without any loss of functionality.

- Shared security
- : The total number of applications and validators in the ecosystem should increase the overall security of all the applications in it through security sharing mechanisms.
- Audibility with privacy
- : The application logic should be auditable by other applications interoperating with it without having to breach the privacy of the application users.
- Business-friendly
- : The solutions should make it easier for Institutional users to utilize decentralized solutions, like non-volatile payment mechanisms, user access control, private chains, key management, and ownership transfer.
- Simplicity
- : The rules and Spec should be simple enough for easy understanding and adoption for a wide range of applications without unnecessary complications or exceptions and corollaries.

2. Footnotes¶

This section defines a few key terms and concepts which will be utilized in the proceeding section for the initiation of the reader. The reader may skip this section if they are familiar with the terms defined here. All definitions under this section are general and no new concepts are introduced in the definitions.

- [1]
- distributed system
- : A system that has its components split over multiple computing machines/nodes that can communicate over a network. The components of the system may or may not be symmetric/serve the same purpose. An app split over a backend, a frontend, and a client is an example of a distributed system with asymmetric components, while a backend system with multiple load-balanced nodes would be an example of a distributed system with symmetric components.
- [2]
- consensus mechanism
- : A mechanism to elect a leader/coordinator to orchestrate a task in a distributed system with symmetric components. A node elected by this mechanism by all the participating nodes may then assign the task to be processed by each node and collate the results when all the nodes have processed and submitted the result of their tasks. The leader/coordinator may be elected periodically or till the current one fails.
- [3]
- crash fault tolerance
- : A distributed system that can tolerate a subset of its nodes crashing/failing disabling them from completing their tasks and continuing operations correctly. A crash fault-tolerant system accounts for its nodes to fail due to factors like network failure, hardware failure, or a system crash and hence not being able to communicate back to the coordinator of the distributed system. Such a system accounts for these failures through the redundant assignment of the same task to more than one of its nodes, so that one node crashing would not lead to the whole system failing.
- [4]
- byzantine fault tolerance
- : A distributed system that can tolerate a subset of its nodes crashing/failing, disabling them to complete their task, or producing incorrect results for the task assigned to them due to system errors or deliberate misreporting with malicious intents. A Byzantine Fault Tolerant system accounts for Crash faults as well as malicious attacks originating from the participating components. Such a system accounts for these failures through the redundant assignment of the same task to more than one of its nodes, with an assumption that the majority of the nodes are not malicious.
- [5]
- distributed ledger
- : A crash fault-tolerant distributed storage system. Any RAID(Redundant Array of Independent Disks) based storage system or a replicated database is an example of a distributed ledger. Distributed Ledgers are also referred to as DLTs(Distributed Ledger Technology). DLTs are best applied in use-cases where a crash fault-tolerant source of information is required for a single application that maintains and utilizes it.
- [6]
- state machine
- : A system that can have a single well-defined state at any given time of a set of possible valid states. The state of the state machine changes discretely when an input is applied to the state machine.
- [7]
- blockchain
- : A Byzantine Fault Tolerant distributed ledger that acts as a periodic state machine with transactions grouped into blocks as inputs to transition its state. While a Distributed Ledger may store its data in any possible format A blockchain stores its data by cryptographically appending blocks into a graph data structure to ensure the immutability of the blocks processed by the Blockchain. Blockchains are best applied in scenarios where more than one applications that do not trust each other maintain access to one source of information. All Blockchains are DLTs but the converse is not true.
- [8]
- dApp
- : Any application that utilizes a Blockchain as a component of its backend. Multiple dApps can run off of one Blockchain. A dApp may have Decentralized components like a frontend, a mobile app, or a backend with business logic. The degree of distributed-ness of dApps may vary based on a case-to-case basis.
- [9]
- PoA
- : A Consensus mechanism where the leader of a distributed system is elected based on their assigned authority to do so. This mechanism is suitable for distributed systems where Byzantine Fault Tolerance is not strictly required, for example for

DLTs.

- [10]
- dPoA
- : A variation of the PoA consensus mechanism where the leader of a distributed system is elected based on the authority delegated to them by the authorized actor in the system.
- [11]
- PoW
- : A Consensus mechanism where the leader of a distributed system is elected based on how quickly the node solves a system-generated cryptographic puzzle. The PoW Consensus mechanism is considered wasteful because of the computational resources it requires to be spent on solving a non-related cryptographic puzzle. Also, the leader election is nondeterministic leading to the splitting of the distributed system into two or more groups if there is a communication delay between the nodes, also referred to as forking. Forking also leads to a non-finalistic application state, necessitating a confirmation period before a state of the distributed system can be considered final.
- [12]
- PoS
- : A Consensus mechanism where the leader of a distributed system is elected based on a minimum limit on their economic stake in the system. The leader election is deterministic in this mechanism leading to faster finality of the system state with no confirmation periods required or forking.
- [13]
- dPoS
- : A variation of the PoS Consensus mechanism where economic stakeholders in a distributed system delegate their responsibility to be elected as a leader to another node in the system.
- [14]
- proposer
- : A node of a Blockchain system that is elected as the leader of the system and has to select transactions to create a block and communicate/propose it to the other participants of the system for validation. The proposer may be incentivized for proposing a block correctly. The proposer may choose to pick up transactions that pay them the most incentive or may even choose to pick no transactions at all for a block.
- [15]
- validator
- : A node of a Blockchain system that is assigned the responsibility of validating the transactions proposed by the Proposer. The validators are incentivized for validating the blocks. A validator when elected can become a proposer.
- [16]
- chain
- : A distributed system consisting of validators as its nodes, with a proposer being elected as its leader based on a specified consensus mechanism with the task of validating transactions to be proposed in a block and appending them cryptographically to a chain of blocks for a blockchain.
- [17]
- stake
- : A representation of economic value on a PoS based blockchain that can be staked to a node participating in the consensus or can be slashed if the node goes faulty or byzantine.
- [18]
- staking
- : The act of delegating the responsibility of participating in consensus, associated with the financial stake held by an actor in a dPoS blockchain system, by binding their financial stake with a validator in the system. The delegator has to carefully choose the validator to stake to as there is a penalty on the total stake incurred if the validator node acts maliciously in the system.
- [19]
- gas
- : A metric of the total work done by the chain to process a transaction sent to it.
- [20]
- transaction fees
- : A representation of economic value on a chain that can be utilized by an application user to pay to the nodes participating in the consensus of the chain to process a transaction. The transaction fees are directly proportional to the gas consumed by the transaction.
- [21]
- node application
- : the application logic being run by a node of a chain.
- [22]
- module
- : a self-contained functionality that can modularly be composed with others to implement anode application
- [22]
- module clique
- : Some modules are dependent on other models for their complete functionality. These modules might further depend on other modules forming a dependency chain. Such a group of modules is termed amodule clique
- of modules and represent the basic minimum set of modules that may be deployed independently as a chain.

4. Architecture¶

The AssetMantle platform is architected as a distributed system with asymmetric components/chains that are run by nodes/validators. The chains run specialized node applications that are built to support different functionalities based on the requirements of dApps that facilitate open finance use cases. The chains interoperate with each other through the Core chain which implements chain interoperability functionalities and orchestrates the validators set and their (dis)incentivization. The dApps can consume the functionalities exposed by the AssetMantle Platform through the AssetMantle client component.

Building and auditing decentralized financial applications is a nascent paradigm with gaps left for failure/errors. AssetMantle implements a common set of well-tested tools that can be utilized to build a wide range of such applications reducing infrastructure risks. Any upgrade/updates/vulnerability fixes to these tools are translated to all the applications built on them. This toolset is implemented across multiple platforms/protocols to make the app-building experience uniform irrespective of the platform/protocol, allowing the flexibility to migrate applications seamlessly between them based on requirement changes. There may be many iterations/versions of these toolsets with some being added/removed.

The AssetMantle ecosystem architecture relies on multiple sovereign chains with shared security and interoperability through a Core chain, the validator set, and uniform application design.

The tech infrastructure is the biggest tool in the AssetMantle application building toolset. It covers the full stack of infra required to get a distributed financial application built and running. The AssetMantle Node Project implements modules to build a custom dApp by stitching the required modules together, while the Client Application implements all centralized toolsets like backend & frontend for the business app, wallets, light clients for the dApp, and explorers. The Core chain connects to all the dApps in the ecosystem and implements all the modules required to allow for interoperability between all the dApps and ensure shared security. The APIs are a set of common API utilities like transaction querying, queuing, ticketing, key management, etc. which allow integration of the dApps with third-party business apps.

4.1. Node¶

extensible, modular, and composable

The functionalities of a chain are defined by thenode application being run by the validators on that chain. Thenode application contains the logic for networking, P2P communication, consensus, Byzantine fault tolerance, and application logic. The AssetMantle Node Project is built in a modular and composable way in contrast to the monolithic approach utilized by the older blockchain implementations. Thenode application is mainly split into two components, the state machine replicator, and the state machine. Tendermint has been utilized as the state machine replicator. It abstracts away the networking, peer-to-peer, and consensus layers into one component. The state machine component encapsulates the state transition logic and is split up into modules that maintain their separate states. The modules may be composed together in several permutations and combinations to produce complex business logic for each chain. Not all nodes running thenode application are validators, the nodes can be classified into:

- Validator Node: A node that can validate transactions and participate in consensus through signing votes. The validator node's main purpose is to validate incoming transactions and optimize compute over storage. A validator node also maintains the validator key. To secure validator nodes from security threats it is run in a militarized zone with no access to the public network.
- Sync node: A node that connects to the chain and keeps in synchrony with the block production without participating in the consensus. The sync node can be maintained for querying the chain state or as a backup node to the validator node.
- Sentry node: A sync node that serves as sentries or bridge nodes from militarized zone to the public network. These nodes connect to all the peers of the chain and bridge them to the validator nodes for consensus participation and validation activities.
- Light Client: A verifier node that does not sync blocks or their history but can query the current state and verify block headers and transaction proofs. The light client is meant for interacting with the chain and for querying or sending transactions to it without participating in consensus or block validation activities.

```ditaa {cmd=true args=["-S"]} +------------------------------------------------------------+ | | | Node | | | | +---------------------------------------------------------+ | | | +-------------------------------------------------------+ | | | | | Cosmos-SDK | | | | | | +--------------+ +--------------+ +-------------+ | | | | | asset | | escrow | | maintainer | | | | | | | +--------------+ +--------------+ +-------------+ | | | | | | | reputation | | negotiation | | ... | | | | | | | +--------------+ +--------------+ +-------------+ | | | | | +-------------------------------------------------------+ | | | | | | | | Application(State Machine) | | | | | | | +-------------------------------------------------------+ | | | | | Cosmos-SDK | | | | | | +--------------+ +--------------+ +-------------+ | | | | | | | auth | mint | | staking | | | | | | | +--------------+ +--------------+ +-------------+ | | | | | | | bank | | ibc | | ... | | | | | | | +--------------+ +--------------+ +-------------+ | | | | | +-------------------------------------------------------+ | | | +--------------------------------------------------------+ | | ^ | | | | | | RPC | | | | -+--+--------------------+---+---+-------------------+--+--+ | | | | | Consensus | | Snapshot | | | | | | | +--------------------+ +-------------------+ | | | | | | Mempool | | Info | | | | | | | +--------------------+ +-------------------+ | | | | | | ABCI | | | | | +----------------------------------------------------+ | | | | | | | Tendermint(State Machine Replicator) | | | | | | | | +-----------------------------------------------------+ | | | | | Tendermint Core | | | | | | +------------------------------------------------+ + | | | | | | Consensus | | | | | | | +------------------------------------------------+ | | | | | | | Networking | | | | +--+--+--+-------------------------+-----------------------+--+--+--+ ^ | | RPC | +---------------------------------+--------------------------------+ | Node(Peer) | | +----------------------------------------------------------+ | | | "" | | | +----------------------------------------------------------+ | +----------------------------------------------------------+

**Fig 4.1** Schematic representation of a Node on AssetMantle Platform

#### 4.1.2 State Machine

- **Cosmos SDK** The Cosmos SDK defines a set of Modules required for DPoS functionalities on top of the PoS consensus mechanism offered by Tendermint. These include logic for delegating, node incentivization/dis-incentivization mechanism, minting new tokens for incentivizing, token transfer, and interchain communication. The AssetMantle Node project imports these modules as is while adding the AssetMantle Modules on top to create the complete app functionality. The modules are composable and can be added/removed from an application based on their use case.
```

- **AssetMantle Node Modules** A collection of a common set of modules that facilitates building dApps for a range of open finance and decentralized exchange applications. All the dApps built using the AssetMantle Node Modules will be natively interoperable with each other through the Core chain. The modules will be also implemented across different protocols/platforms/languages to allow inter-platform native interoperability between the applications and similar application behavior/make the application platform/language agnostic and easy to migrate across them based on requirement changes. These are some of the modules implemented by the AssetMantle Module set (this set may change in the future):

- AssetFactory: create, mutate, and burn NFT tokens. This module can be utilized for use-cases that require tokenization of assets/issuing digital assets/certificates.

- AssetWallet: send an asset to other account wallets. This module can be used if the asset ownership is exchangeable.

- reputation: keep a count of successful transactions and allow the counterparty to set feedback against a transaction.

- contract: negotiate and record terms of an exchange transaction.

- escrow: escrowed exchange transactions that are bound by a contract.

- maintainer: access control and hierarchical access to transactions. This module can be utilized for a use case where there is an issuer organization or an organization with a hierarchy and division of labor.

4.1.1 State Machine Replication

- **Tendermint Core:** A Byzantine Fault Tolerant state machine replication mechanism. The Tendermint Core is responsible for the networking, peer-to-peer gossip, and consensus layers of the node application. It implements a Proof of Stake consensus mechanism to elect a proposer for chain blocks.

- **ABCI:** A generic interface that the application layer has to implement to interact with the Tendermint Core functionalities. The ABCI is split into four connections for consensus, mempool, info, and snapshot. The consensus connection is a collection of methods that are responsible for consensus-related application call and block execution logic, mempool has methods to validate incoming transactions into the mempool, info has methods to query the application state and snapshot has methods to maintain and restore state snapshots. An application has to implement these methods to connect to the Tendermint Core consensus network and hence become a blockchain.

Tendermint core and ABCI together form the Tendermint state replication machine with the four connections types exposed either over a network as RPC or can be compiled together with the application into one binary if the application is written in golang. The application may be written in any language and maintained and run as a separate binary along with the tendermint binary interoperating through RPCs over the network.

4.2. Chain

A chain is an asymmetric component composing the AssetMantle Platform distributed system. A chain has many nodes/validators which are tasked with validating the node application logic assigned to the chain. Each chain has different security, fault tolerance, trustlessness, and decentralization requirement based on the application it runs. The AssetMantle Platform allows the chains to have a different number of validators, types of consensus mechanisms, governance structures, and validator incentivization mechanisms based on the chain requirements.

```ditaa {cmd=true args=["-S"]} +------------------------------------------------------------------------+ | | | Chain | | | | +-----------------------------------------------------------+ | | | Validator | | | | +-----------------+ +-----------------+ +----------------+ | | | | | KYC {d}| | Compliance{d}| | Governance{mo}| | | | | +-----------------+ +------------+ +----------------+ | | | | | | | | +----------------------------------------------------------+ | | | | | Node | | | | +--+--------------------------------------------------+--+ | | ^ | | | | | | RPC | | | | | +--------------------------------+-------------------------------+ | | | Validator(Peer) | | | | +----------------------------------------------------+ | | | | | "" | | | | | +----------------------------------------------------------+ | | | +----------------------------------------------------------+ | +------------------------------------------------------------------------+

Fig 4.2 Schematic representation of a Chain on the AssetMantle Platform

#### 4.2.1. Validators¶

The Validators on the AssetMantle Platform are entities that operate nodes to validate the platform chain's application logic. The complete set of all the validators on the platform operates on the Core chain. The Application chains have a subset of the Core validator set operating on them. The shared shed of validators facilitates shared security and (dis)incentivization mechanism on the platform based on the consensus mechanisms in operation on the Application chain. The Expectations from a validator on the platform are:

- Validate the code base for inconsistencies/errors/backdoor into the application logic.
- Validate the new proposed smart contracts to run on the dApp for stable operation.
- Validate on the Core chain.
- Validate on the application chains if required, submit evidence of Byzantine faults at the dApp, and adhere to any legal requirements that may be necessitated by the operator business of the application.
- The node operator may have to undergo a KYC process to participate as a validator in some of the Application chains based on the business application requirements.
- The node operator may have to agree to abide by legal compliance and SLA to participate as a validator in some of the
```

Application chains based on the business application requirements.
- All the validators have to validate on the Core chain and participate in its governance.

4.2.2 Consensus Mechanism¶

The consensus mechanism of a chain is decided based on the governance and economic model of the chain's business application, the security and fault tolerance level required, and the resource feasibility of the chain. The consensus mechanism used on the AssetMantle Platform are modifications of the base PoS consensus mechanism offered by the Tendermint Core with changes applied at the application layer.

- DPoS:
- A chain that has its definition of economic stake and requires Byzantine Fault tolerance has to implement the DPoS consensus mechanism for block proposer node election. The dPoS consensus mechanism is the default on the Core chain.
- PoA:
- A chain that doesn't have an economic stake definition that can be slashed/minted and doesn't require Byzantine Fault tolerance has to implement the POA consensus mechanism for block proposer node election. The validator nodes can be authorized directly by the chain operators to validate.
- DPoA:
- A chain that doesn't have an economic stake definition that can be slashed/minted but still does require Byzantine Fault Tolerance has to implement the DPoA consensus mechanism. In this mechanism, the operators of the participating validator nodes have to bond a stake on the Core chain to be deemed eligible to validate on the application chain. The Core chain, based on the boded stake of the validator node then authorizes the nodes to validate on the application chain. The incentivization of the nodes happens at the application level and the dis-incentivization in case a node behaves maliciously is disincentivized on the Core chain.

4.2.3. Governance¶

The Bitcoin blockchain application utilizes a PoW consensus mechanism, that produces cryptographic puzzles that a node has to solve to become the leader/proposer of the next block. The complexity of the cryptographic puzzle is not arbitrary, it's a parameter that is decided by Bitcoin's PoW consensus mechanism based on the performance of the nodes of the chain in the mining time of the preceding blocks. Similarly, there are consensus parameters that need to be modified in a PoS consensus mechanism too, and these parameters are much more numerous in the case of DPoS. These parameters are changed by a mechanism which is termed chain governance, where a node can propose a change to the parameter, and the rest of the nodes vote to either accept or reject the parameter change proposal.

On the AssetMantle Platform, each chain has its governance mechanism based on the flavor of the PoS consensus mechanism utilized in a chain. Each module exposes a list of parameters too that can be changed by the governance of the chain. It is the responsibility of the validator nodes to participate and vote on all governance proposals. The parameters are designed to change incrementally and have a cool-off period so that the chain could not change too drastically too quickly.

4.2.4. Validator Incentivization¶

All the chains including the Core chain on the AssetMantle platform implement mechanisms to incentivize the validators, with the dis-incentivization part being handled by the consensus mechanism. There are two main categories of incentivization mechanisms on any AssetMantle platform chain:

- Block Rewards:
- Every block that is added to a chain produces block rewards. This reward serves the purpose of incentivizing validators to maintain chain liveliness when there are no transactions to propose on a given block. The block reward is calculated as a proportion of the stake that a participant has staked into the system. The stake may or may not have economic value based on some of the application chains, with then only representing the authority metric to validate on the chain.
- Transaction Fees:
- Every transaction that is added to a block needs to be paid for by the originator of the transaction as transaction fees. The transaction fees serve the purpose of prioritization of transactions and setting limits on the work that the chain needs to perform to validate one transaction. The transaction fees are calculated as a function of the computation operations required by a node to process the transaction, plus some extra offered to prioritize the transaction(bribe to block proposer?). The fees may or may not have direct economic value but may rather be a function of multiple factors that are defined by the business application logic of the chain.

4.3. Platform¶

ditaa {cmd=true args=["-S"]} +--+ | | | AssetMantle Platform | | | | +------------------------------------+ | | | +---------------+ +---------------+ | | | | | Validator A | | Validator B | | | | | +---------------+ +---------------+ | | | | Application A Chain | | | +----------------------------+----------+ | | | | | | IBC Protocol | | | | | +--+---------------+--+-------+-------+--------------------+ | | | | Validator A | | Validator B | +---------------+ | | | | +---------------+ +---------------+ | Validator C | | | | | +---------------+ | | | | | | | | AssetMantle Core chain | | | | | | | + +---------------+ | | | | | Validator D | +---------------+ +---------------+ | | | | +---------------+ | Validator E | | Validator F | | | | +--------------------+-------+-------+--+---------------+--+ | | | | | | IBC Protocol | | | | | +----------+----------------------------+ | | | +---------------+---------------+ | | | | | Validator E | | Validator F | | | | | +---------------+ +---------------+ | | | | Application B Chain | | | +------------------------------------+ | +--+

Fig 4.3 Schematic representation of the AssetMantle Platform

4.3.1. Core¶

The Core chain connects to all the other chains in the ecosystem and implements interoperability transactions with them. It has the complete set of the Validators participating on the platform validating on it, making it the most secure chain and ideal for the third party escrowed/moderated transactions. The Core's node application implements mechanisms to incentivize or subsidize a Validator set for participating in the Application chains and decentralizing them if they misbehave on these applications through governance proposals and shared security mechanisms.

The Core chain is at the center of all of the governance activities in the AssetMantle ecosystem, handling proposals for platform level governance like adding Application chains, whitelisting and blacklisting validators to validate at the Application chains, validator stake slashing if they misbehave maliciously at the Application chain, with more proposals being added with Core chain improvements.

The Core chain's node application implements modules that are interoperability-focused, like wallets and escrows. All the Application chain's application logic can interoperate with the Core's and Application chain's application logic for inter-application token ownership exchange.

Repository AssetMantleCore

4.3.2 Mantle¶

The Mantle chains are application chains that implement amodule clique . These chains are the base implementations of the AssetMantle node modules which are meant to be operated as test net for sandboxing new module functionality before putting them in in-production chains.

Repository assetMantle

4.3.3 Crust¶

The Application chains implement custom business logic through implementing a combination of AssetMantle Modules and may have legal/SLA requirements for the Validator set. The Application chains connect to the Core chain not only for interoperability with other chains but also for shared security and governance. The Application chains may also not be multi-tenant or open to public validation.

Repository comdexCrust

4.3.3. Shared Security¶

Inter-chain Staking & Governance

The Core chain has a definition of an economic staking token baked into it, but not all dApps running on the AssetMantle platform will necessarily have to define one of their own. But a chain operating with a PoS consensus mechanism can only defend against Bayzentine Faults only if the stake offered as a security by the Validators has an economic value. The Core chain staking tokens serve the purpose of providing this value to be staked as security for other Application chains.

A new chain is added to the AssetMantle platform through a governance proposal created by the new Application chain operators on the Core chain. The Validators on the Core chain have to vote on the proposal to get the new Application chain added to the platform. The Application chain operator defines the other specifics of the chain like the consensus mechanism to be used, the max validator set size, the minimum stake on the Core chain to propose to become a validator on the Application chain, the KYC and legal compliance requirements for the validator and the Application chain's validator incentivization mechanism.

The Core chain validators propose to become the new Application chain's validator if they fulfill the operator requirements and have staked the required tokens. If more than the minimum required set of Validators is required to propose to validate on the Application chain then the top validators, by the amount they staked on the Core, are picked. The stake is locked in for the validation period of the validator and servers as security for the new Application chain created.

In the scenario that a validator acts maliciously or fails to comply with the Application's SLA requirements, the Application chain operators or the Application chain validators raise a governance proposal with evidence of the event. The Core chain validators then evaluate the evidence and vote on the proposal to remove the validator from the Application chain and slash the stake of the Validator.

If the validator makes it to the end of the proposed validation period on the Application chain, the tokens staked by them are released and sent back to the validator's operator wallet. This mechanism allows a new chain to start operations without having to worry about onboarding a reputable and distributed validator set or having to introduce a new staking token and making sure that its value is retained above the value of the transactions secured by the chain (defend against the nothing-at-stake problem).

4.3.5. Specification¶

The AssetMantle ecosystem depends on many native data structure implementations to define its application logic. To allow for native interoperability between all the applications in the ecosystem different implementations of these data structures have to adhere to a common specification defined by the AssetMantleSpec . Some of theseSpec being:

- interNFT

- : an NFT interface specification.
- interIdentity
- : a specification for a user identifier on the chain.
- classification
- : a specification for defining any type or property set of a data structure.

4.3.5. Integrations¶

The functionality of some dApps may depend on external APIs and data feeds for their complete functionality. AssetMantle implements these integrations as PoA oracles sending cryptographically signed API messages to the dApps. The centralized APIs have to be modified at the provider's end to add the cryptographic signing and verification functionalities. These integrations can then be utilized at any dApp in the ecosystem. Some of these integrations are:

- Fiat Ledger
- : a layer 2 on top of the banking layer that allows for tracking of incoming and outgoing fiat banking transactions through Bank API integrations. The Fiat tokens created through this process then act as stable coins in the ecosystem and can be utilized for payments. The value of these tokens can then be redeemed back for the fiat value through outgoingFiat Ledger
- integrations.
- Identity
- : an integration with KYC and credential providers for creating an identity of the user in the ecosystem.

4.4. Client¶

A collection of dApp client tools that connect with the chains and help build business/client-facing logic on top of them. The client tools are built in a generic and modular fashion to allow for easy adoption into any use case and quick deployment with dApp integration out of the box. The client tools are updated across applications and with each SDK module update or for a vulnerability fix. These client tools can be white-labeled for new applications/use cases.

4.4.1. Business Application¶

A complete client-facing application with a Backend, Frontend, and AssetMantle Node module functionality integration. The business client is built in a modular fashion to allow flexibility in client app flows. The business client can be white-labeled for new use cases. The project is maintained privately.

4.4.2. APIs¶

The APIs are a set of API endpoint accessibility tools that help make help third-party client app integrations with dApp and help make the RPC endpoints easier to consume. Some of the APIs tools:

- Transaction ticketing
- : A real-time application generally fluctuates in TPS and requires an interface to manage the transactions to other rate-limited systems, like the Blockchain application. Current blockchain applications have been built with direct interaction with the end-user in mind with the user handling all the failures and transaction rate limiting/recovery. Thetransaction ticketing
- tool helps rate limit the transactions and handle error scenarios and retries.
- Signer
- : A light client that maintains the private keys of the end-users and signs transactions sent to it on their behalf.signer
- accepts transactions in a simple JSON format and makes the private key custody and recovery easier for the end-user.
- Key Management
- Transaction Queuing

4.4.3. Web Integration Libraries¶

A third-party business application can integrate and use the AssetMantle platform's dApp service by integrating with them through the Web Integration Libraries. These libraries provide integration components to consume dApp functionalities like sending transactions and querying their status, transaction success notification, and app and chain state querying.

- Wallet:
- Queries the user wallet's contents to display the current Fungible Token balances, NFT balances and handle metadata representation, and integrates to make payments and transfers in apps with relevant denominations.
- Transaction Generation:
- Accepts transaction input data from the integrating application and marshalls it into a transaction object compliant with the dApp's light client format.
- Signing:
- Accepts transaction object and Keystore with passphrase as input to generate a signed transaction. The integrating application can then relay the signed transaction to the dApp's light client.
- Notification:
- A subscriber client for the events published by the blockchain which converts and notifies the user about events relevant to them. The utility can be integrated at the backend of a business application too to keep its schema synced with its dApp state.

RepositoryAssetMantleJS

5. Discussion¶

This section discusses the emergent features of the architecture defined in this document.

5.1. Scaling & Sovereignty¶

In a Byzantine Fault Tolerant distributed system, the participant nodes are assigned individual functions with some nodes replacing the tasks to ensure that the system performs normally even if some of the nodes crash or start acting maliciously. The level of replication may depend on how error-prone the task is or the expectation of the number of malicious actors in the system. The AssetMantle Platform is such a distributed system with a validator serving as the nodes and the tasks being the dApps being run on the platform. The number of validators replicating the state of a dApp on the platform is decided as a function of the crash and byzantine fault expected from the dApp. This approach is in contrast to the Blockchain 2.0 chains like Ethereum where all the dApps are replicated across all the participant nodes irrespective of the security level required for that particular dApp. This leads to reduced transaction processing speeds and rate limitations based on the transaction load being experienced by the other dApps running on the chain.

The defined architecture allows for each dApp to operate sovereignty, isolated from the environmental factors affected by other dApps running in the platform, allowing for linear scaling of transaction speeds as the number of dApps grows.

5.2. Business-Facing vs. Customer-Facing Approach¶

The AssetMantle platform architecture has been designed for utilities and optimizations for a business-facing dApp. This approach is opposed to the majority of the current generation dApp which are designed to directly face the end customer with their schema optimized for human comprehension and error handling left to them the end user. The customer-facing approach allows for a greater level of trustlessness as the end-user does not have virtually any centralized components to confuse the dApp services. The user may not even have to depend on a third-party app to interface with the dApp, by directly interfacing through a command line. But this approach, unfortunately, suffers from three drawbacks:

- The steep learning curve associated with each new dApp
- The cognitive limitations of the user to understand the dApp evens and be able to respond to Ethereum
- The end-user being error-prone will make mistakes that have to be handled by the users themselves, making the dApp service consumption a potentially very time-intensive activity
- The assumption is that all business processes can be transformed into totally decentralized processes with no intervention of a third business actor on top of the user and validator actors

The trust requirement in a business process can be minimized but to make it trustless might not be possible with the immediate business environment and even if possible might make the system too arcane to interact with, hinder adoption and infringe on user privacy. The AssetMantle Architecture has been defined to be friendly to Business Processes with flexible schema definition to allow for easy integration. The interfaces have been designed for interaction with Business applications with a focus on a unified schema definition for the communication rather than human readability. This allows the dApps to be easily consumed by the Applications that require trustless interactions with other participants' applications and can package the services into a human-friendly format. The Business operator is responsible for the legal and ethical liabilities of operating the application while the dApp serves as a persistent and immutable information management system with trustless authentication and authorization through cryptography.

5.3. Auditability & Verifiability¶

A centrally administered MIS system that has to comply with audit requirements generally has to maintain audit logs which are the system logs with the requests and responses made to the system recorded in a plain text format. Such a mechanism is required to facilitate an external private audit of the system and ensure that the system is neither faulty nor are there any Byzantine actors in the system. This approach suffers from some major drawbacks:

- The logging is up to the implementation of the application and not a defined standard
- The logs are not easily verifiable for errors or malicious actors
- The external auditors are exposed to the private data
- The auditors may themselves be Byzantine and produce false audit reports
- Even in the case of an honest audit process the faulty component or the malicious actor is exposed well after the event has happened with the error having multiplied and spread to other systems as well
- In the case of a dishonest audit, there is no way for an external actor to verify the audit reports with the system only being exposed only after the error has multiplied to a size notable by external actors, or in the majority of the cases, they might never even be exposed
- The audit process is only valued against the trust and reputation of the auditor with external actors, leading to the centralization of trust with a few reputed auditors, who may drive up the audit cost as the demand for audits increases

The defined architecture reduces the trust required between a customer and the auditor actors instead of aiming to remove the trust requirement between two customers as the main objective. The validators in this scenario act as the auditors, that keep on auditing the app at runtime for the publicly visible data. To conserve privacy some of the data in a schema may not be directly exposed but rather be exposed as a hash with the real data being stored with a Business Application. When this business application is being audited, the audit logs data can be verified against the hash of the data and processing timestamp recorded on-chain. This structures the audit process and validation for the private data through data verifiability.

5.4. Resource Optimization¶

The Byzantine and Crash Fault-Tolerant system achieve their tolerance through replication of tasks on multiple nodes. This implies that the cost to perform a task also increases with the level of tolerance offered by the system. A different application may require different levels of fault tolerance. The described platform allows flexibility with the level of fault tolerance a dApp might need by segregating them to separate sovereign chains, instead of running all of the apps in one chain at the highest security level possible sacrificing scalability. This leads to resource optimization based on the level of security required for a business use-case.

- Cost:
- The transaction fees incentivizes a proposer to pick a transaction when the transaction mempool is larger than the transactions that can fit in block size. In one chain for all dApps approach if a business depends on a mission-critical transaction to be processed they will have to spend extra fees to get a transaction processed. Other dApps that do not produce enough economic output to justify the transaction fees may be indefinitely halted in such scenarios. All the dApps should not depend on all the nodes to replicate the transaction. The described architecture allows a dApp to define the level of replication and fault tolerance required while segregating from other dApp operation influences. This allows for reduced transaction processing costs with predictable
- Compute:
- All the nodes of the proposed distributed system engage in separate tasks with limited replication with no compute being spent on the wasteful leader election process. This brings down the compute resources required to validate a dApp down to basic server hardware with no dependence on complex hardware solutions like ASICs and graphic cards with 100s of processor cores.
- Storage:
- Storage is one of the most critical resources in single-chain multi dApp applications since the 1 Byte of storage used by the app translates to 1*N bytes effectively used on all the nodes in such a system. By reducing the amount of replication in the system, the proposed architecture reduces the storage requirements for running a dApp allowing dApps to store a lot more data immutably on the chain itself.

5.5. Identity & Credential Management¶

One of the biggest blockers to the mass adoption of blockchain has been the trivial but risk-prone task of cryptographic key management. This has prevented numerous use cases like the usage of blockchain for payments at a mass level or institutional adoption for asset management. One of the key innovations of the AssetMantle Node architecture is the replacement of cryptographic keys as identity with an issued identity system, called interIdentity.

InterIdentity allows each dApp to issue identities to the users, that serve as usernames for web applications. These identities are interoperable across the platform making it possible for other dApps to accept them as a user identity too instead of onboarding the users afresh, like OAuth. This enables the users to maintain a single identity across dApps giving a standard definition that is accepted across these dApps.

InterIdentity also dissociates cryptographic keys as a source of identity and instead uses them as credentials for an identity that is replaceable. Each identity can be linked to multiple keys and each can be used on the behalf of the identity to sign transactions. Each device can generate a non-exportable key that the user can link with their identity to send transactions from that device. In the case of loss of the device and hence the keys, the user can then disassociate the key from their Identity, eliminating the threat of it being misused. InterIdentity makes identity and credential management a lot more simplified for the general and institutional users.

5.6. Modularity¶

The proposed architecture is modular in nature with the networking, P2P, consensus, and node modules split into separate layers that can be composed together to form a node application. This is in contrast with the node architecture of legacy blockchain node architecture offering better reusability and composability of the components.

- Blockchain 1.0: Bitcoin, where all the layers are composed into one binary with an upgrade requiring a fork to the chain. In the AssetMantle platform, an update of a layer like a consensus, P2P, networking, or application can be handled by a node updating the binary and restarting, with the node missing a few blocks but with no forking. If the changes between the two versions of Tendermint or the Application logic being updated are breaking changes then through governance a fixed block height is decided where all the nodes halt operation, update the binaries and restart their nodes back up, with a few nodes missing some of the blocks in the process.
- Blockchain 2.0: Ethereum, where the application is a monolith but runs a virtual machine where new application logic can be plugged in and all the nodes of the distributed system have to validate all the application logic symmetrically and hence defend against Byzantine Faults. On the AssetMantle platform, the distributed system has multiple asymmetric components performing different application logic computations, with each component/chain having some level of redundant node computations to avoid Byzantine Faults. The number of these nodes performing the redundant tasks may vary based on the potential risk of the component/chain being exposed to Byzantine or Crash faults.

6. Source¶

The latest version of this paper is maintained at the AssetMantleOne githubrepository .

7. References¶

1. Satoshi Nakamoto, 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf
2. Libp2p, 2020. Technical Documentation. Retrieved from https://docs.libp2p.io/
3. Jae Kwon, 2014. Tendermint: Consensus without Mining. Retrieved from https://tendermint.com/static/docs/tendermint.pdf
4. Jae Kwon, Ethan Buchman, 2018.Cosmos: A Network of Distributed Ledgers. Retrieved from https://cosmos.network/resources/whitepaper

Previous Technical Terms Next Terminology