presents

okto

Whitepaper v1

Introducing a Fully Expressive Orchestration Layer

This draft whitepaper invites feedback from the community and co-builders. You can reach out to okto-whitepaper@coindcx.com for feedback and collaboration.

Contents

Background

Web3 tech/innovation and community have been revolutionary in so many ways; that said, adoption is still picking up, slowly but steadily. For web3 to be all-pervasive and touch every tech stack in the world (in a major or a minor way), we need to bring web3 to the users and developers, not users and developers to web3.

Over the last several years, this industry has solved fairly conclusively for scalability and fundamental infra primitives. Modularity, via composability and open permissionless systems, has always been a core tenet of blockchains, but now we see that charged even more. Not only are we seeing an explosion of app chains, but modular architecture allows the best of the best to emerge without overly relying on one monolithic solution. This 'modularity movement' has recently picked up even more steam with the separation of data, execution, and consensus layers, and we are seeing it in all corners of web3. We can quickly expect 100,000+ app chains and millions of apps, each optimized for specific use cases/experiences, to be the status quo in the near future. While this creates a lot of value, it also amplifies fragmentation challenges. Fragmentation of not only liquidity but also user experiences and technology standards. So, the next immediate problem for mass adoption emerges as the unification of liquidity, experience, and standards across blockchain solutions.

While the first generation of unification solutions mostly solved liquidity fragmentation through various innovative approaches - bridging solutions (e.g., Layerzero, Wormhole, etc.), there is a lot of research and work happening in solving it more natively through solutions like Aggregation Layers or OmniChain dApps. Even the unification of user experience has started picking up steam with various high-quality teams building Chain Abstraction solutions (e.g., NEAR). At the same time, we believe that there is a need for an end-to-end solution that solves both developer challenges that come from fragmentation and broken user experiences due to fundamental blockchain primitives. To solve this, we foresee an *'Orchestration Layer'* that abstracts web3 complexities and solves developer/user experiences via solving for the three-pronged challenge of fragmentation (liquidity, tech standards, and user experience).

Such an *Orchestration Layer* is the next layer in the modular tech stack and will be closest to the application developers. On the one hand, it solves chain abstraction. It removes the complexity of standards/chains/different protocols for developers and at the same time, enables them to offer simple and familiar web2-like experiences to end-users (e.g., single-click experiences to access multi-step/complicated web3 primitives).

Summary

The Okto Orchestration Layer (‘Okto’), is a middleware that solves for experience for developers and end-users on the one hand, and for GTM for web3 protocols/chains on the other hand. By enabling it via a chain, provides massive permissionless, decentralized scalability that enables expressiveness and solves for a large number of use cases and experiences.

On the left-hand side of the diagram Fig 1.1 below, are web3 protocols. On Okto’s platform, these protocols create immutable ‘Blocs’ or programmable scripts to enable access to their protocols in a relatively standardized manner. On the right-hand side of the diagram are the application developers who, through a standard set of APIs can access the capabilities exposed by the Okto platform. Okto is the middleware (built as a trustless chain) that does the heavy lifting of abstracting away web3 transaction complexities while creating simple developer and user experiences. Okto solves chain abstraction, asynchronous transaction management, simple use case Blocs (or programmable scripts), delegated signing, and data indexing. This abstracts nuances of the different chain ecosystems to provide a consistent developer experience. This enables the developer not only to cut their development time by over 90% but also enables web2-like single-click experiences for their end users; it allows the developer to focus on their core product while the web3 technology/complexities are abstracted away.

Okto achieves this through 3 core components -

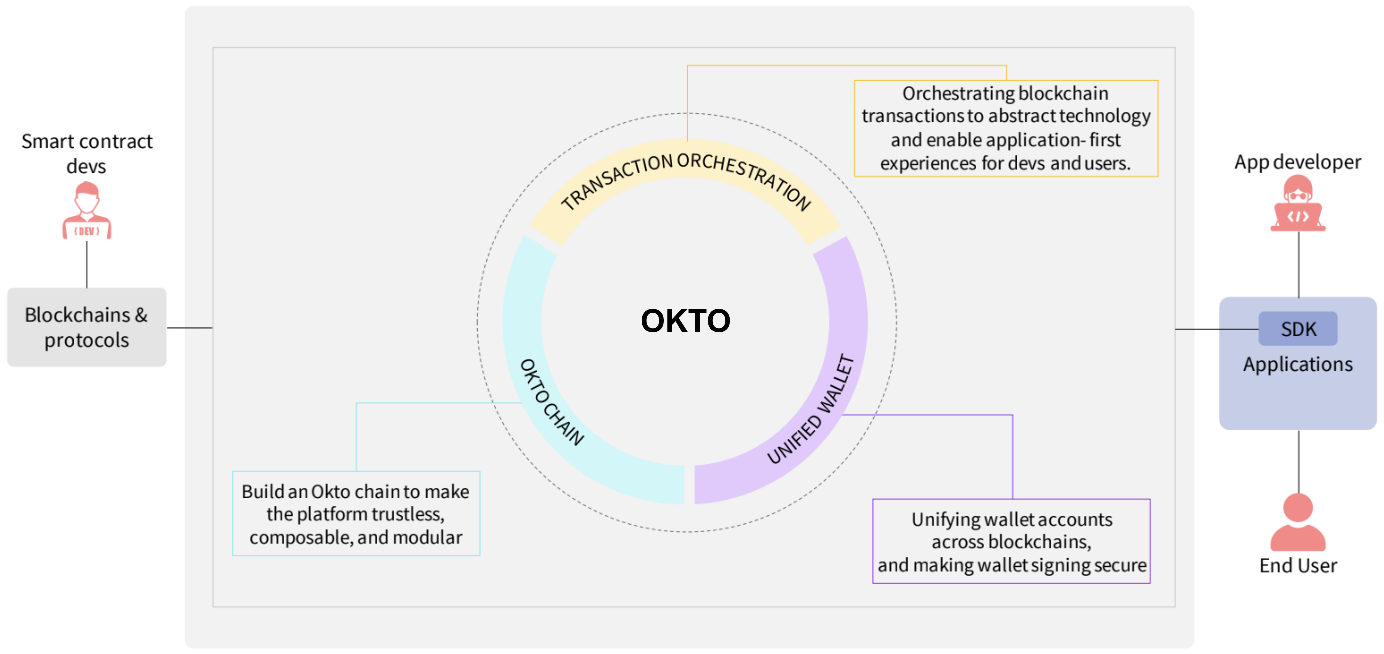
- **The Okto app chain** orchestrates the entire solution. It stores unified user accounts (including permissions etc.), transaction information, set-up information, and protocols’ Blocs. All other components inherit trust from this immutable app chain ledger. We should note that this chain is not a typical user-centric chain that keeps user account assets or has TVL/liquidity; this is a middleware chain that instead contains information required to orchestrate seamless transactions while user assets and liquidity/TVL still lie on the underlying chain the protocol operates on. This is built as a rollup-based app chain, secured and network-incentivized by the $OKTO token. We don’t intend to solve problems already solved, and this chain will be built as a roll-up on one of the existing secure/scalable blockchains to start with, but with required changes that enable seamless orchestration. While there are several components to the chain, the chain has 2 critical external facing sub-components i.e.
  - A **Bloc Hub,** where web3 protocols/chains build and deploy Blocs (or scripts) that enable connectivity into Okto with a simpler set of primitives. Different Blocs could be composed to create new blocs as well.
  - **A unified set of APIs** that app developers can use to enable web3 use cases in their apps while providing familiar/simple user experiences. These simple API primitives are built to abstract and significantly reduce development time/complexity.

- **Decentralized Wallet Networks (*aka* DWNs)** enable *Unified Wallet Accounts*, which are MPC-secured, and enable permissions/delegation through the app chain’s ledger. The user delegates wallets on this network to sign on their behalf based on the permissions provided. This enables delegated flows across chains that support Account Aggregation (EVM ecosystems) and even for those with more straightforward EOA/UTXO or other such account constructs on ecosystems like BTC, SOL, and MOVE-based ecosystems like APTOS/SUI, etc.

- **Decentralized Transaction Networks (*aka* DTNs)** perform asynchronous *Transaction Orchestration* across blockchains and protocols. Transaction management provides a chain-abstracted, gas-abstracted, protocol-abstracted, transaction-lifecycle-abstracted experience to developers and their end-users. For example, a developer will not have to manage multiple steps inside a simple transaction like nonce management, gas-fee estimation, failures and retries, asset assertions, data indexing, etc. Such management, when done across chain ecosystems and standards, becomes unmanageable, and the DTN solves it with asynchronous state management.

The DWN and DTN are built as off-chain actor-networks since transaction orchestration requires asynchronous management across chains, which is currently not supported by the virtual machines of any mature chain ecosystem. They are built as network actor nodes that are still secured via a decentralized approach of nodes incentivized and secured by the crypto-economic security of the native token $OKTO.

Fig 1.1 below illustrates what we are building with various network stakeholders.

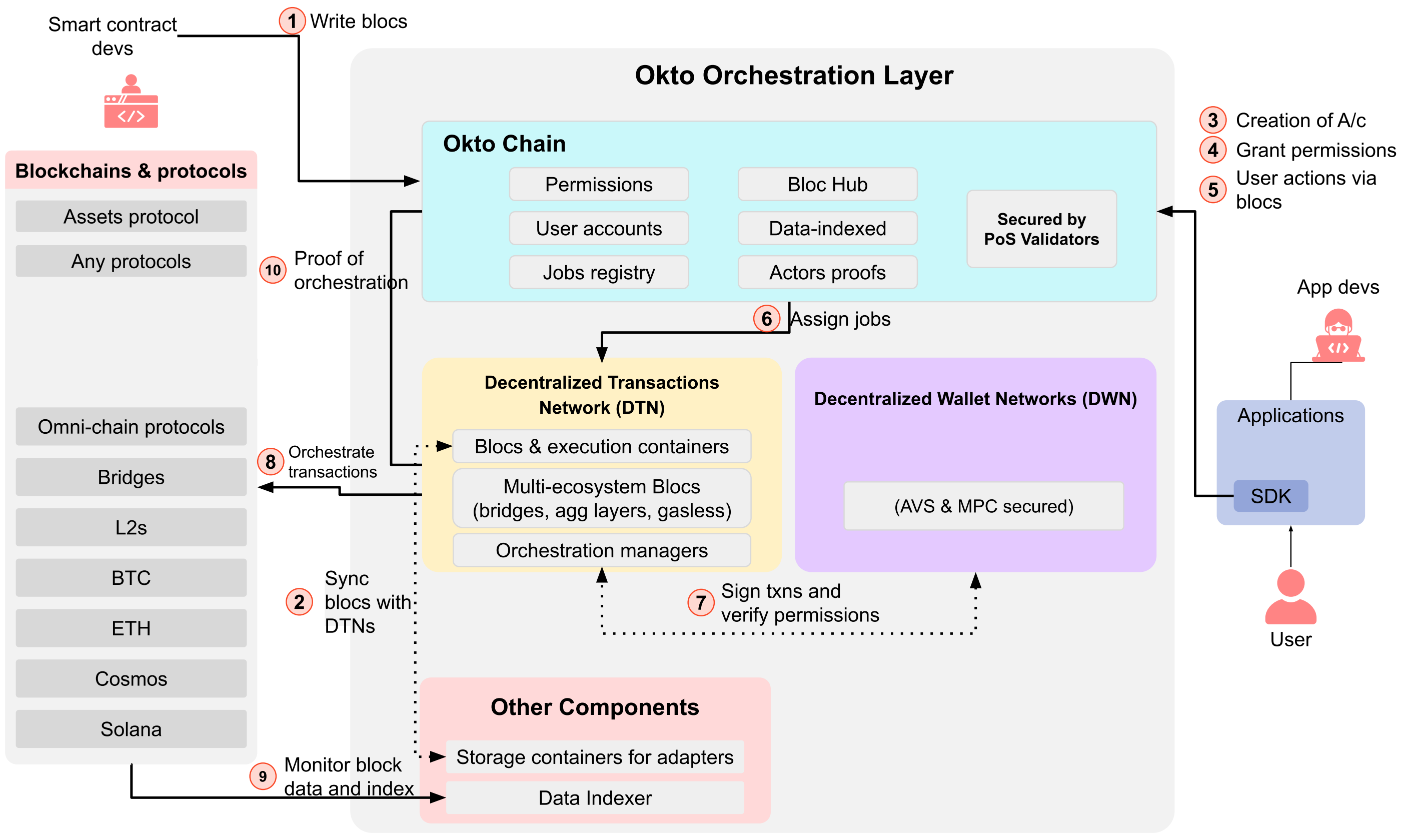


Smart contract devs
DEV
Blockchains & protocols
TRANSACTION ORCHESTRATION
Orchestrating blockchain transactions to abstract technology and enable application- first experiences for devs and users.
OKTO
OKTO CHAIN
UNIFIED WALLET
Build an Okto chain to make the platform trustless, composable, and modular
Unifying wallet accounts across blockchains, and making wallet signing secure
App developer
SDK
Applications
End User

Building the Okto Orchestration Layer

Okto is designed to orchestrate web3 use cases by leveraging three critical components: the *Okto app chain*, *Decentralized Transaction Networks (DTNs), and Decentralized Wallet Networks (DWNs)*. These layers work together to orchestrate simple APIs for developers, effectively becoming a middleware orchestrator to the blockchain ecosystem. Fig 1.2 provides an overview of the architecture.

Fig 1.2: Okto Orchestration Layer, an Architecture overview



Platform Components

Okto app chain

The Okto chain is unique because it is not a typical chain that holds user assets/TVL. The chain is middleware built specifically to orchestrate transactions that abstract away web3 complexity and simplify developer experience. The chain doesn't compete with any L1, L2, or L3 app chains today for scalability or TVL; its primary purpose is to help applications build with simpler primitives and better user experiences than what we have in web3 today. Since we don't intend to build a new VM for now, we will partner and leverage the ongoing work/research within the L1/L2 ecosystem and build this as an app chain on an existing L1/L2.

The Okto app chain works with network nodes in DWN and DTNs as the orchestration required to simplify a user action to multiple sub-transactions across chains requires asynchronous transaction management. A provable VM architecture is early in research and development and will take some time to mature. To solve the problem today, we propose the app chain plus decentralized node model, which provides security while maintaining the ecosystem's delegated self-custody ethos. This platform enables secure delegated flows within the ecosystem and yet allows asynchronous end-to-end transaction management.

In the following sections, we take an example where a use case developer writes Blocs on our Bloc Hub to support the management of SoulBound NFTs on any chain built using the Okto primitives.

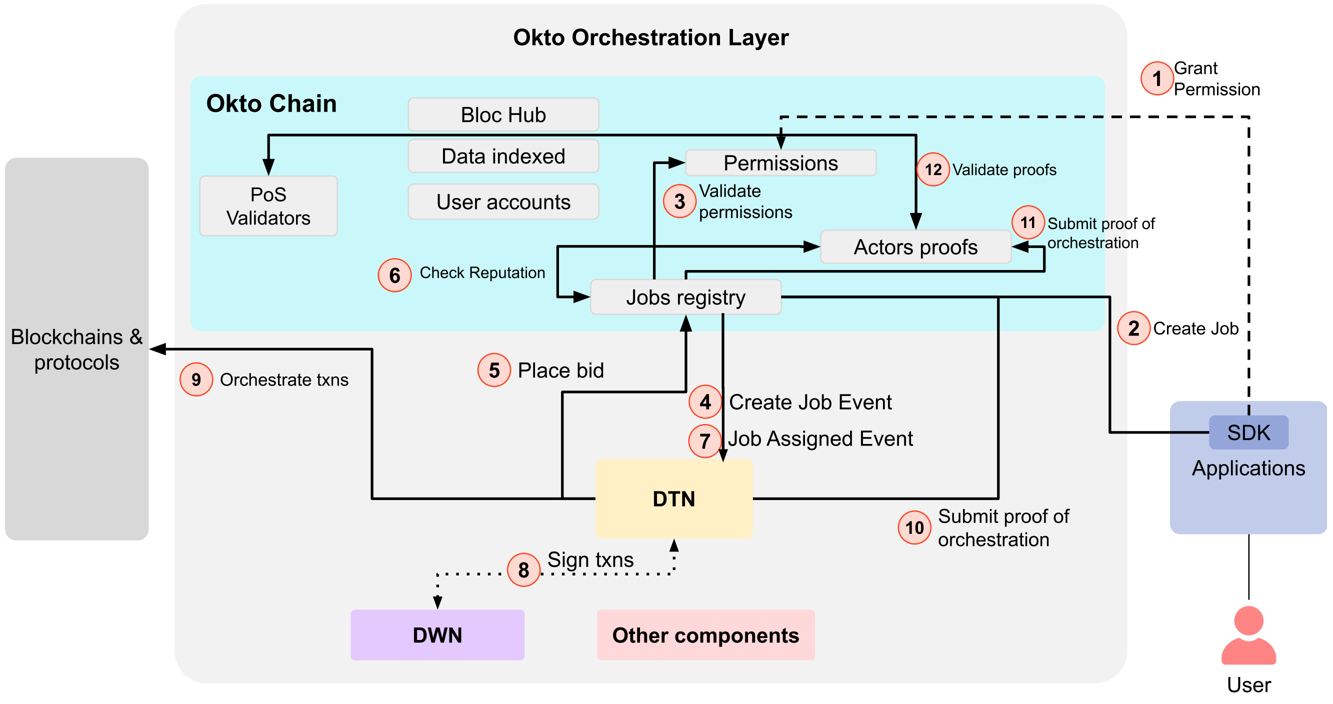
Set up steps before a transaction is enabled via the Okto platform

1) A smart contract developer writes a Bloc using the Okto primitives and syncs with the Bloc-Hub.
2) An application developer refers to the primitive of this Bloc within their apps through the Okto SDK.
3) An app's end user gets a unified wallet on Okto using any social OAuth, and grants delegated on-chain permissions to the application using Okto SDK.

Fig 1.3 shows the steps involved in the following transaction.

4) An application developer calls 'CreateJob' on OktoSDK with relevant Bloc parameters, in this case, 'CreateNFT,' to create a Soul Bound NFT containing user details.
5) Smart contracts on the Okto chain create a job and raise it to the DTN nodes. The nodes bid with their best execution cost, and the winning bid node receives the job to execute it.
6) The smart contract on Okto selects the best DTN node based on cost and reputation.
7) The smart contract on Okto waits for the completion of the job and pays the execution fees.
8) Data Indexers update user account data with the new Soul Bound NFT data.
9) DTN Validators verify the proof of orchestration.

Fig 1.3: How the Okto app chain works



The app chain ledger contains the following contracts in their registries:

1. **User Account registries**
   a. **User accounts:** *maintains objects storing the details of the users' wallets and its ephemeral keys controlling the account.*
   b. **Account permissions:** *maintains a set of permissions for chains, protocols, assets, and asset quantities delegated to application sessions.*
   c. **Indexed data:** *maintain a list of user transactions and their target transaction details.*

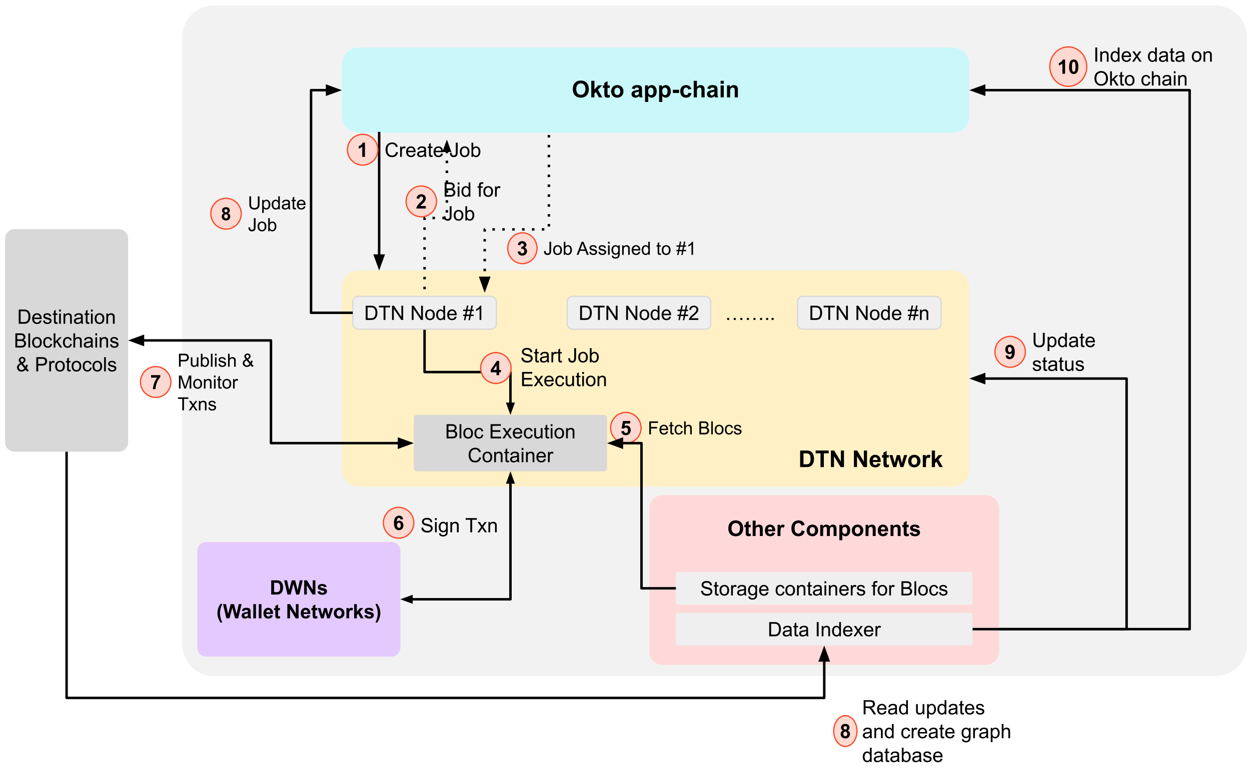
2. **Bloc registry**
    a. **Bloc registry:** *maintains a registry of use case programmable scripts (Blocs), their functional parameters, and required permissions; developed by web3 developers.*
    b. ***Simple APIs for Blocs:*** *maintains the simple APIs exposed via SDKs to application developers.*
3. **Crypto-economic security registries:**
    a. **DTN node data**: *maintains a list of all the node details, staked amount, their reputation, and the jobs being performed.*
    b. **DWN node data**: *maintains a list of all the node details, staked amount, their reputation, and the session permissions for signing.*
    c. **Proofs**: *maintains the ZKProofs from the validators for successfully executing the jobs by the network nodes.*
    d. **Validator nodes & PoS contracts**: *Maintains the staking contracts to ensure the nodes' crypto-economic incentives.*
4. **Okto Treasury:** manages the treasury, crypto-economic security, revenue shares for the network participants, etc.

Decentralized Transaction Networks (DTNs)

The DTNs are responsible for orchestrating end-to-end transactions. They do the heavy lifting and perform the required sub-transactions for the intended user action. The user action required might require multiple approvals, transactions, data aggregation, and gas-management across multiple chain ecosystems: all these sub-transactions are handled by the DTNs and the Okto app chain to deliver the end user outcome.

They are set up as decentralized nodes (inspired by Layer0's DVNs, Chainlink DONs, etc.); any user action is broken down and executed. This transaction orchestration is enabled via decentralized node models and is not directly a part of the app chain VM. The validator-set would verify the proof of script execution, and the transaction hashes finality at destination chains. These transaction flows require asynchronous interactions with chains/protocols to provide a better/seamless user experience, and current chain VMs do not provide for such asynchronous flows. We plan to evolve DTN nodes into provable VMs where the scripts and their execution can be verified externally using ZK proofs and then sent for signing by DWNs.

Fig 1.4: How Decentralized Transaction Networks (DTNs) work



1. **Bloc Resolution [BR]**: Each node contains a registry of all possible Blocs. The node uses the appropriate use case Blocs to create jobs and create a set of instructions for the participating blockchains in the job. The Blocs cover all asset classes, and the community can submit more Blocs as mentioned in later sections.

   Eg., *the Job contains the Bloc hash, which needs to be executed along with its functional arguments. The Bloc will create a SoulBoundNFT on the target blockchain for the user. The arguments contain the list of metadata to be added for the NFT. The execution container resolves the Bloc hash for the IPFS and its related dependencies and executes it inside an execution container.*
2. **Asset Assertion [AA]**: The node creates a list of all the assets affected by the job; assets can be across any blockchain ecosystem. The node utilizes the Data Availability Layer to check if the wallet account has the asset and gas on the requested chain.

   Eg., *as part of the Bloc, the initial step is to ensure the gas exists for the creation of the NFTs, asset assertions take care of the gas, and if any ERC20 tokens are required for the creation of NFTs.*
3. **Asset Consolidation [AC]**: Based on the assets affected, *Bloc* resolution nodes create a set of additional instructions that contain the best possible path for consolidating the assets on the target chain. These instructions use a pool of possible paths like decentralized bridges, DEXes, etc.

   Eg., *asset consolidation will ensure native tokens in the target chain are made available from a list of source chains using a set of best possible decentralized bridges.*
4. **Transaction Creation [TC]**: Nodes create a list of all the transactions to be published and use the appropriate double-spending prevention and cost optimization logic according to the ecosystem's requirements.

   Eg., *the Bloc will create a MintNFT transaction payload written by the Bloc developer which will be executed on the target chain. Once the payload is created it can be sent for signatures and transaction lifecycle management.*
5. **Transaction Signing [TS] and Publishing [TP]**: DTN nodes request DWNs to sign the list of transactions created. DWNs perform independent checks of the transactions and wallet account session permissions and sign them. Nodes relay each transaction sequentially on all the destination chains and manage acceptance in

the blocks.

Eg., *this is the core of the Bloc, where the platform and the DTN executor nodes will ensure gas, nonce, and signature management. Once the transaction is signed, it is published to blockchain RPC nodes, and the DTNs ensure that the transaction is accepted in the blockchain sequencer. A failsafe-based retry approach is executed if the transaction is dropped for any reason.*

6. **Data Indexer[DI]:** DI responsible for listening to each block on every blockchain supported and filtering the events associated with Okto users' accounts. These events are processed, and data is indexed for every wallet account.

   Eg., *The newly updated soul-bound NFT would be indexed by the data indexer which creates graph-indexed data for the user account and can be accessed by the end-user application directly.*

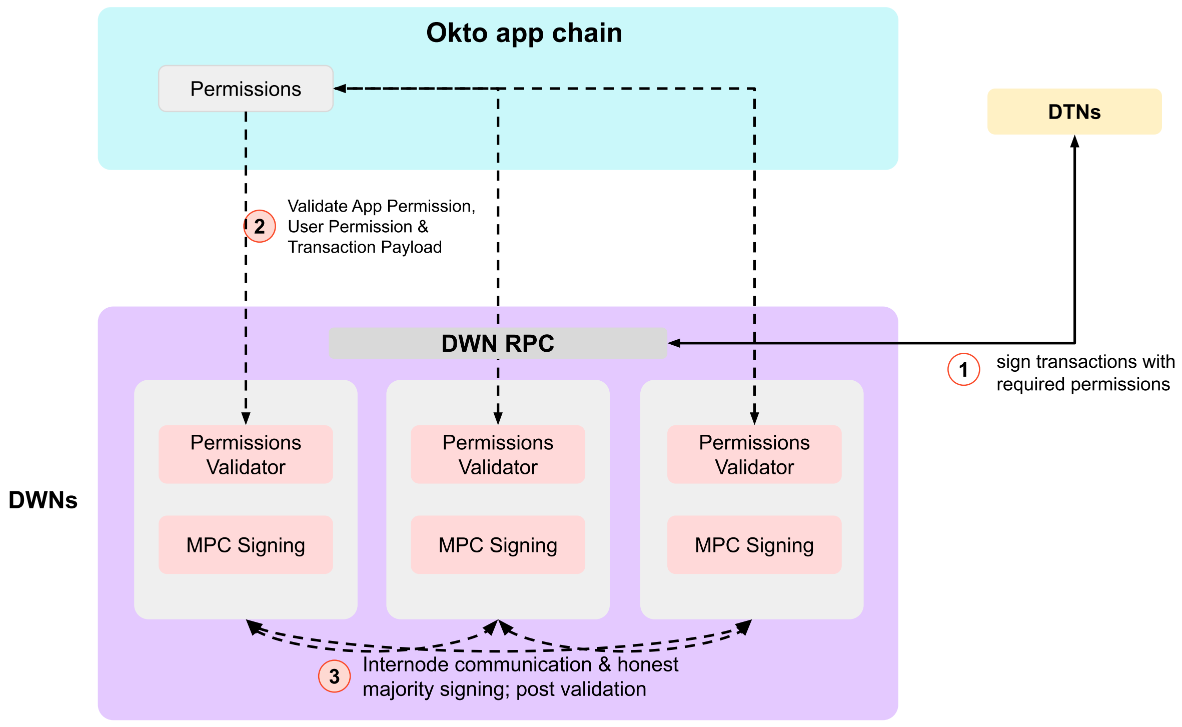
Decentralized Wallet Networks (DWNs)

The Decentralized Wallet Network (DWN) runs a set of delegated MPC signers with permissions to perform actions on the smart contract wallet or act as independent delegated EOAs for the users. DWN is a set of decentralized MPC nodes that consult with the app-chain registry of security and session policies controlled by end-users to mitigate their risk. These nodes are decentralized by a set of off-chain independent nodes hosted by Actively Validated Services (AVS) nodes of EigenLayer; which independently bid, sign a transaction, and get rewarded in $OKTO as fees. This model is inspired by the concept used by web3 auth to keep the experience server orchestrated, hence simplified, yet retain trust-minimized control of signing. Fig 1.5 illustrates the process.

Because of this architecture of DWNs, the Okto platform can support EVM, non-EVM, and even potential private chains, and orchestrate transactions across all of them. This is made possible by the delegated MPC wallets across all these chains unified as a user account on Okto. Users enable on-chain permissions via Okto client SDKs for various applications via app-session-risk policies. These session policies are stored in the permissions registry on the Okto app chain. DWN works closely with DTNs to sign transactions based on permissions on the Okto chain, and asynchronously performs multiple sub-transactions. This architecture/approach thus enables simplified delegated user flows that enable single-click experiences for multi-hop/chain/protocol transactions.

Eg., *continuing the NFT example, the DTNs created a functioning payload, but DWNs will give the go-ahead for the payload to be signed and published to destination chains. The DWN RPC node gets the complete payload and signatures generated from user session keys present at the end-user application. RPC relays the payloads to all the DWN nodes. Each actor-node has a two-part scheme: one verifies the payload's content relayed to networks and two if the session keys contain the required permissions to sign the payload. Each DWN node verifies this independently by checking the app-chain ledger for permissions. Once verified, the nodes participate in the signing operation for their MPC node. If most nodes achieve consensus independently on this signing process, the fully signed payload is returned to the DTN for publishing.*

Fig 1.5 How the DWNs work



Okto app chain
Permissions
DTNs
2
Validate App Permission, User Permission & Transaction Payload
DWN RPC
1
sign transactions with required permissions
DWNs
Permissions Validator
MPC Signing
Permissions Validator
MPC Signing
Permissions Validator
MPC Signing
3
Internode communication & honest majority signing; post validation

Bloc Hub

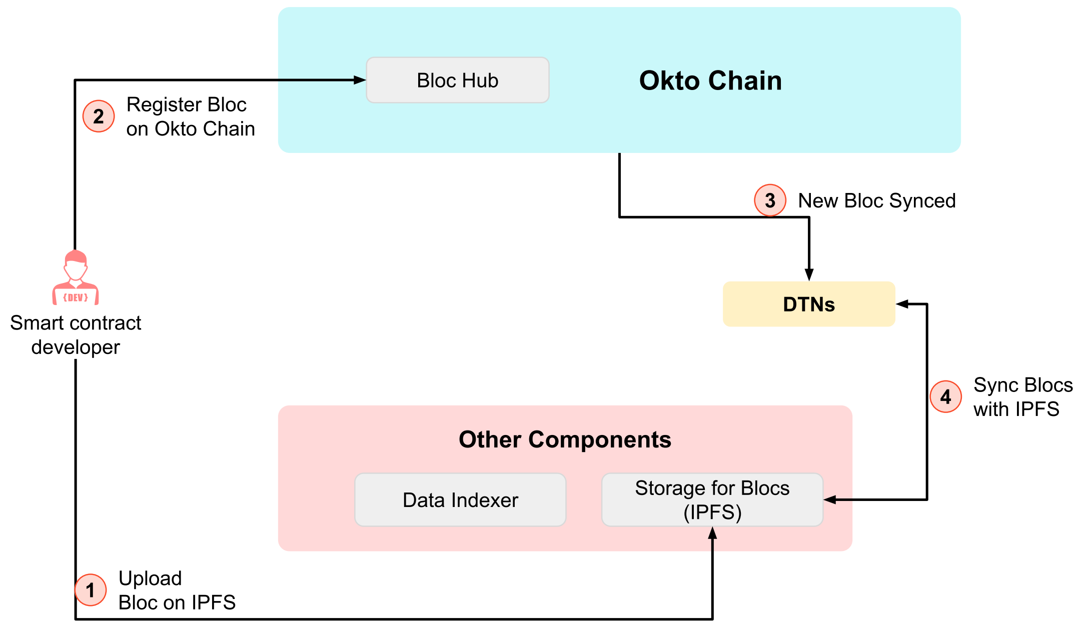
The Okto team has built several natively supported Blocs (or scripts) for DeFi, GameFi, and NFT use cases. However, new use cases in web3 will keep growing exponentially, and native Blocs won't be able to keep up. With this composable architecture, the community can develop Blocs for new use cases and contribute to the Hub. Bloc developers will be incentivized in the $OKTO token as per the on-chain utilization of their Bloc. As new blockchain ecosystems and innovations increase, developers can innovate by creating new Bloc types and leveraging primitives to build new use cases and support new ecosystems. Platform Blocs can be built in leading programming languages like TypeScript, etc.

This creates a programmable platform that abstracts chains, wallets, and assets while building a great user experience. The Bloc hub is extensible and made trustless via the app chain ledger to cater to app developers' orchestration and indexer needs. Once a use case is registered and approved by the community, it is available to all the nodes of DTNs, and application developers can access them using the API SDKs. This integration streamlines development, reducing dev time significantly and attracting application developers to integrate protocols in a composable manner. Ultimately, users benefit from expanded access to protocols alongside an enhanced user experience.

Eg., *the web3 developer can also improve the SoulBoundNFT Blocs to create an ENS corresponding to the avatar, the developer would write TypeScript Blocs using the primitives available and upload the Blocs onto IPFS using the Okto API SDKs. Once uploaded and approved the Blocs hash, functional requirements, and permissions requirements are added to the Blocs registry of the Okto app chain. Any new addition to the Blocs registry mandates the DTNs to sync and verify the uploaded Blocs within their execution environment. Bloc's reputations are also maintained in the app chain and regularly screened by the community.*

The use case Blocs can be as complex as payment networks similar to Gnosis Pay - where the blocs can consolidate the assets across various networks and have programmable delivery of a user action.

Fig 1.7: How the Bloc Hub works



Security of Okto chain

Okto's platform's architecture follows the principles of decentralization and trust-minimized security, which are fundamental to this *Orchestration Layer*. Okto is an app chain built on an existing L1/L2 and inherits the security of the existing underlying consensus layer.

It further enhances security using Independent nodes, crypto-economic incentives, on-chain registries, and policies. By decentralizing its operations across independent nodes, and leveraging on-chain registries for risk mitigation policies, the platform ensures that no single entity controls the system. This structure mitigates central points of failure and promotes a secure, transparent environment for transactions and interactions.

Security via Independent Nodes & Crypto-Economic Incentives

The DTNs use node-based security with inspiration from LayerZero DVNs, Chainlink's DONs these nodes will be extended into a two-part security model: one, on proof of execution where the DTN will produce a proof of execution against the scripts; and second, a set of transaction hashes produced as part of the job. Using these points the validator network will verify the integrity and finality of the jobs.

DWNs are MPC-based nodes that use DKLS and honest majority-based security inspired by Torus Network and DFNS. DWNs are hosted by Eigen Layer AVS nodes, which provide security to Ethereum validators using re-staking.

Data Indexers are composed of independent nodes that operate based on crypto-economic incentives like Graph Protocol.

A set of validator nodes safeguard the above three systems against bad actors.

On-Chain Registries and Policies

On-chain registries manage policies for Blocs, and wallet accounts maintain the platform's trustlessness. The access to on-chain jobs and accounts is based on ephemeral key-based session models, which provide delegated sessions to each end-user application with narrow permissions. These registries ensure that all operations within the platform are transparent and verifiable by anyone in the network. It allows users to confidently interact with the system, knowing that their transactions are processed based on immutable rules without censorship.

This architecture ensures that each node is incentivized to perform their tasks accurately and efficiently, contributing to the network's overall reliability and security. Finer details of the crypto-economic incentives via the $OKTO token will be covered in the v2 release of this whitepaper.

Progress & Roadmap

Milestone	Status	Timeline	Details
Launch of Okto Platform	Completed	Q2 2023	● Launch of Okto Platform with native cross-chain bridges. ● Integration of 20+ EVM, Cosmos, Aptos, and Solana chains. ● Launch of transaction bundling and gasless transactions on all supported chains.
Launch of Okto Wallets	Completed	Q2 2023	● Launch of Okto client Wallets leveraging the Okto platform ● Launch Asset Class Scripts (NFT, Tokens) ● DeFi scripts for Swap, Futures, Passive Earn
Launch of Okto Platform Clients	Completed	Q1-Q2 2024	● Launch of Okto Lite Wallet SDK leveraging the Okto platform ● Integrate with CoinDCX ● Launch with 30+ app builders
Launch of Okto app chain	In Progress	Q3 2024	● Launch of Okto app chain. ● Launch of DWNs on AVS nodes and wallet registries on Okto app-chain ● Launch of an beta version of DTN, allowing assets management using Okto app-chain. ● Release of $OKTO Tokenomics details. ● Users can create an Okto chain account from any existing client app.
Extension of Okto - DTNs	Planned	Q4 2024	● DTNs extensible for custom Bloc deployment and available to use from client apps. ● Launch of beta-cohort of Bloc builders, and their Blocs will be available in client applications.
Permissionless Blocs	Planned	Q2 2025	● Builders will be able to deploy permissionless Blocs on Okto.
Decentralized & censorship-resistant	Planned	Q4 2025	● Launch of provable DTNs, which will provide proof of orchestration. ● Launch of DTN validator network ● Decentralization of DTN and Data Indexers actors.

Tokenomics for the $OKTO token will be detailed in the next version of this whitepaper. The $OKTO token aligns network incentives for security and bootstrapping the application network. To bootstrap the network, an initial 7% of the token allocation is reserved for early adopters, allocated through Okto Points.

SDKs of the Okto Orchestration Layer

Okto's API SDKs are available to developers, such as the Okto Lite embedded wallet SDK for app developers, and a Bloc SDK for protocols to build their own Blocs. These API SDKs enable developers to build use cases from the Bloc Hub on Okto. The DeFi API SDK simplifies DeFi use cases like swaps, derivatives, lending, and passive-earn for CEX-like clients, first launched with CoinDCX. These SDKs are available to developers unfamiliar with blockchain complexities, and would like to use the embedded SDK primitives to build their web3 application.

Some sample primitives from the Okto Bloc SDK are given below, which protocols can use to build their own Blocs.

1. sign(hash): Primitive to sign any hash and generate a signature
2. send(chain, transactionObject) ➝ jobId: Primitive to submit any transaction object on a targeted chain
3. consolidateAssets(chain, asset, quantity) ➝ jobId: Primitive to consolidate asset to a target chain with the desired quantity
4. waitForConfirmation(jobId): transactionHash -> primitive to wait and retry till block confirmation
5. assertFunds(chain, transactionObject) -> boolean: primitive to assert funds availability
6. read(chain, contract, functionArgs): bytes ➝ primitive to read from any supported chain
7. externalScriptCall(scriptHash, arguments, metadata): bytes ➝ primitive to call any external bloc within the platform.

Team

This whitepaper is presented by the CoinDCX group. The CoinDCX group was last valued at $2.15B in 2022 having raised over $240M+ in funding. The CoinDCX group houses the CoinDCX CEX (the largest in India with 15M+ users), Okto ecosystem, and partner CEXs in the Middle East, and Africa. The company is backed by some of the most prominent web3 investors (Polychain, Block.one, Pantera, Coinbase, Draper Dragon, Cadenza, Kindred) and web2 investors (BCap, Steadview, Kingsway amongst others).

Appendix:

Okto-Platform Experience: A Case Study

The platform's potential to revolutionize blockchain interaction is best illustrated through a practical example of a GameFi application developed on this platform. This case study showcases the seamless user experience and highlights the ease and satisfaction developers experience when building on this platform.

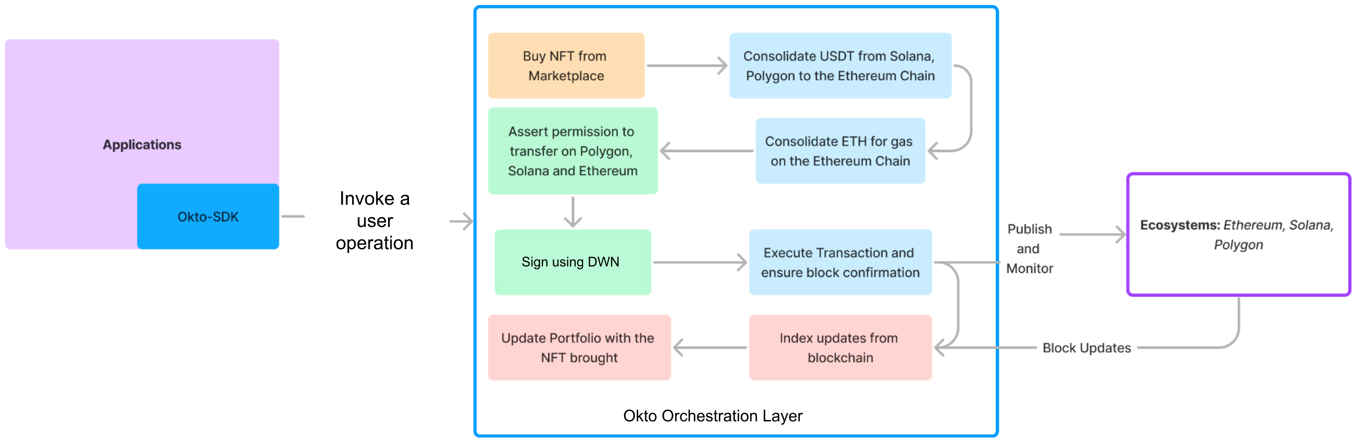
Developer Journey: Building a GameFi Application on the Okto platform

Initial Concept and Development:

A team of developers conceptualizes a GameFi application incorporating blockchain technology to offer a unique gaming experience. The game involves earning tokens through gameplay, which can be used for in-game purchases, and collecting digital badges representing players' achievements and reputations.

Using Okto for Blockchain Interactions:

The developers leverage Blocs to implement token transactions and badge assignments within their game. The ease of integrating these blockchain elements using the Okto platform's APIs significantly reduces the development time and complexity. Okto's DTN simplifies the interaction process between the blockchain and the game's backend.


Applications
Okto-SDK
Invoke a user operation
Buy NFT from Marketplace
Consolidate USDT from Solana, Polygon to the Ethereum Chain
Assert permission to transfer on Polygon, Solana and Ethereum
Consolidate ETH for gas on the Ethereum Chain
Sign using DWN
Execute Transaction and ensure block confirmation
Update Portfolio with the NFT brought
Index updates from blockchain
Okto Orchestration Layer
Publish and Monitor
Ecosystems: Ethereum, Solana, Polygon
Block Updates

Launch & Growth:

The game launches to acclaim from the blockchain gaming community. Developers continue to utilize the Okto platform for new features, such as cross-chain functionality, allowing users to bring assets from other blockchains into the game seamlessly. The game's success is a testament to the robust, developer-friendly platform we can offer.

User Journey: Engaging with the GameFi Application

Onboarding and Asset Management:

A player, new to blockchain gaming, finds onboarding a breeze with the DWNs. They use social login options and grant delegated asset access, allowing them to start playing without navigating the complex world of blockchain wallets and transactions.

Earning and Using Tokens:

As the player engages with the game, they earn tokens for their achievements. Thanks to seamless asset consolidation and transaction processes, these tokens are easily managed and spent within the game for various in-game items and advantages.

Collecting and Flaunting Digital Badges:
The player also collects digital badges, which serve as a reputation system within the game. These badges, stored securely on the blockchain, are a source of pride and a testament to the player’s skills and achievements. The Data Indexer ensures these assets are easily viewable and verifiable, enhancing the player’s sense of accomplishment and ownership.

Cross-Chain Interactions:
Eventually, the player explores using assets from other blockchain ecosystems within the game. Okto’s chain-agnostic framework facilitates this, allowing the player to experience true interoperability and the richness of a multi-chain universe without technical hurdles.

References

1. https://panteracapital.com/blockchain-letter/blockchains-dial-up-to-broadband-moment/
2. https://www.forbes.com/sites/williamanderson/2023/01/18/the-three-hurdles-holding-back-blockchains-mainstream-adoption/?sh=24e1f5241588
3. https://www.bankless.com/2024-is-the-year-of-modular-expansion
4. https://twitter.com/drakefjustin/status/1754554466609635483
5. https://a16zcrypto.com/stateofcrypto/
6. https://blog.availproject.org/the-avail-vision-accelerating-the-unification-of-web3/
7. https://layerzero.network/publications/LayerZero_Whitepaper_V2.pdf
8. https://polygon.technology/blog/aggregated-blockchains-a-new-thesis
9. https://axelar.network/
10. https://wormhole.com/
11. https://dappos.gitbook.io/docs/infrastructure-design/how-dappos-works
12. https://www.biconomy.io/post/modular-session-keys
13. https://www.firechain.io/docs/architecture-guide
14. https://twitter.com/firechain/status/1580253525379948546
15. https://eips.ethereum.org/EIPS/eip-5192
16. https://docs.ipfs.tech/concepts/further-reading/academic-papers/#ipfs-content-addressed-versioned-p2p-file-system
17. https://docs.zksync.io/zk-stack/concepts/hyperchains-hyperscaling.html
18. https://docs.polygon.technology/cdk/
19. https://docs.optimism.io/stack/components
20. https://www.dfns.co/
21. https://www.eigenlayer.xyz/
22. https://eprint.iacr.org/2023/765.pdf
23. https://www.bankless.com/2024-is-the-year-of-modular-expansion?ref=bankless.ghost.io
24. https://chain.link/whitepaper
25. https://github.com/graphprotocol/research/blob/master/papers/whitepaper/the-graph-whitepaper.pdf
26. https://docs.tor.us/key-infrastructure/role-of-torus-nodes/overview
27. https://medium.com/layerzero-official/layerzero-v2-explaining-dvns-02e08cce4e80
28. https://polygon.technology/blog/layer-2-demystified-how-polygon-scales-ethereum