Chainlink

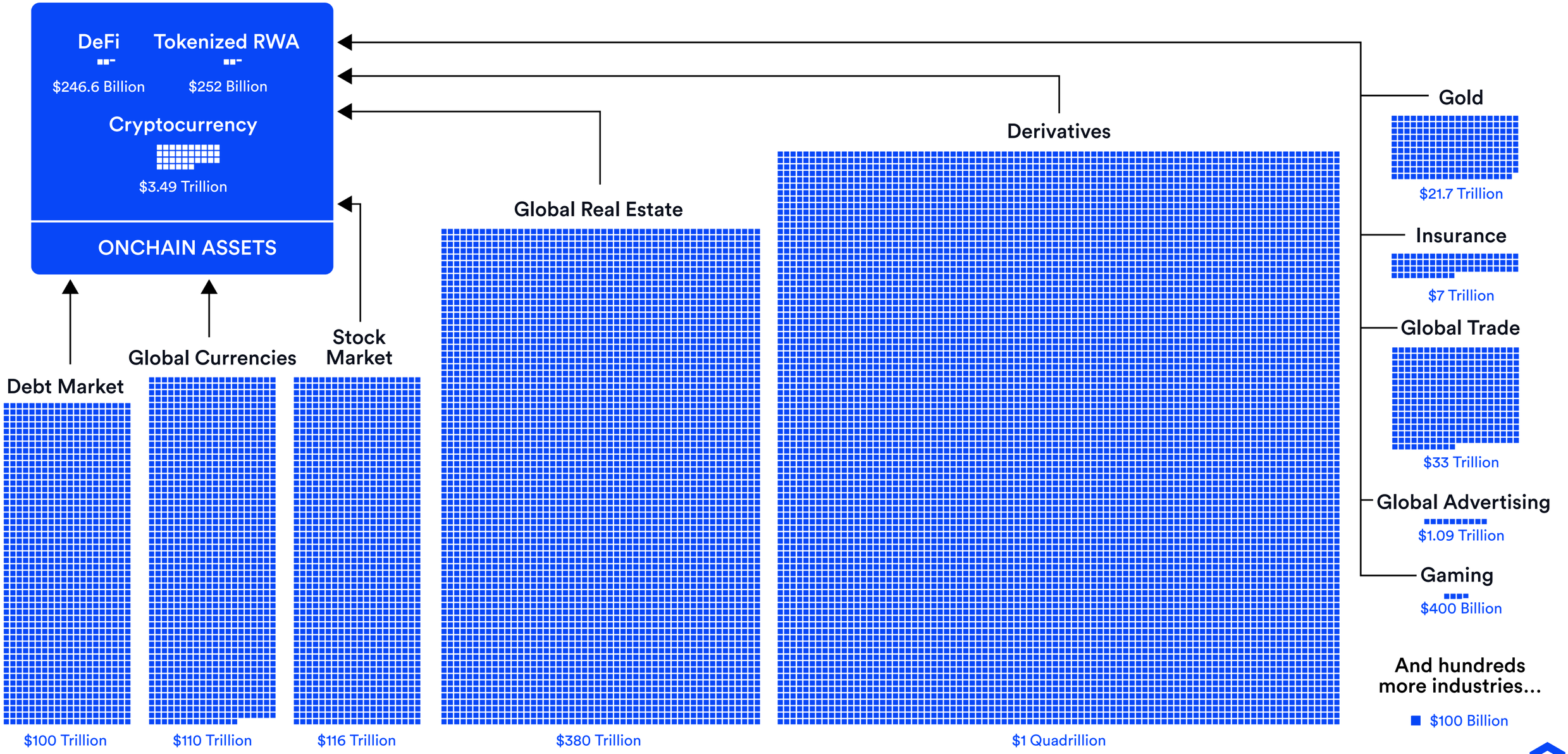
Building the Next Generation of the Web3 Industry

Consensus 2025

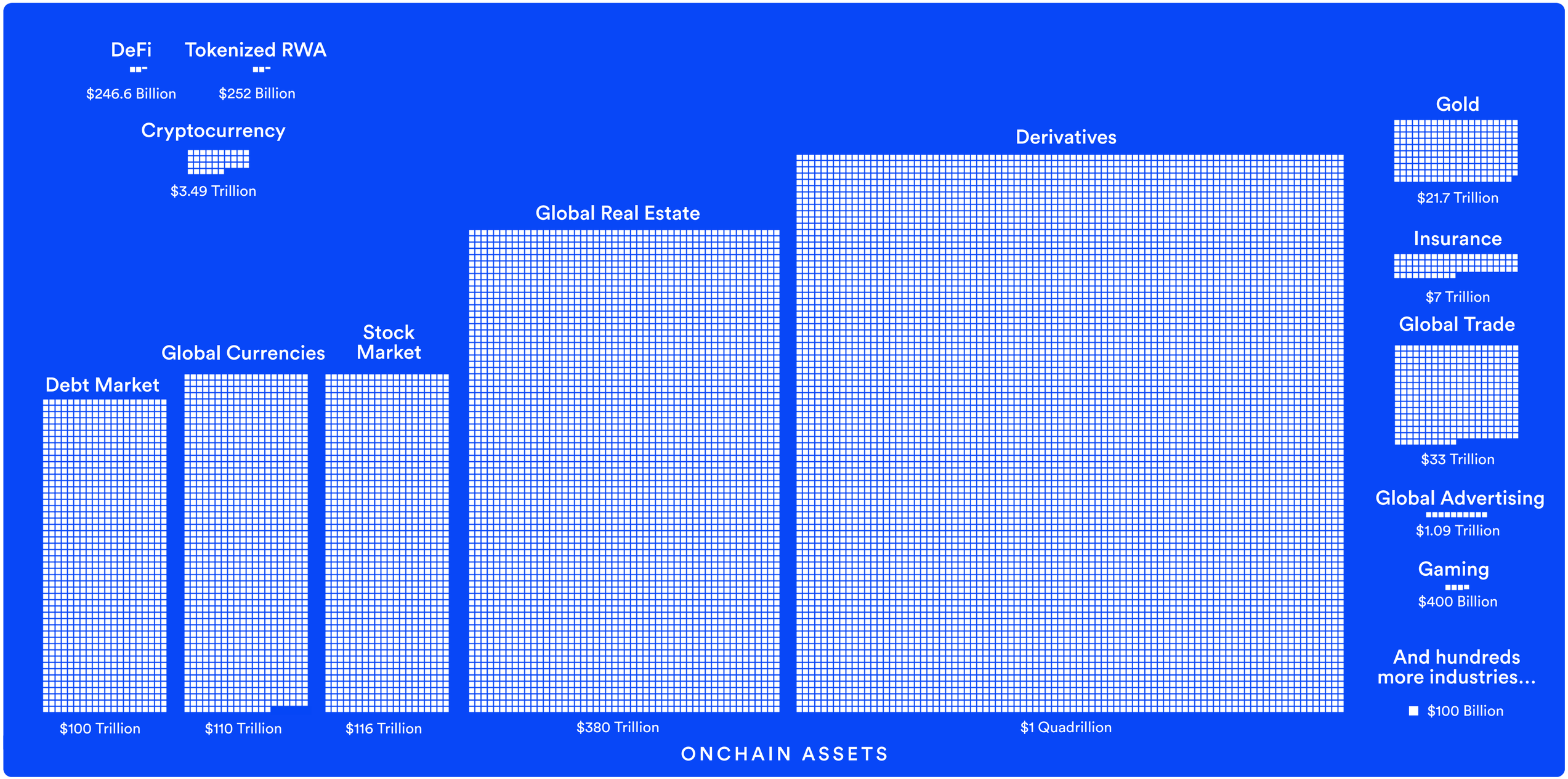
The Key Drivers of the Web3 Industry's Growth

- RWAs like stablecoins for payment and tokenized funds as assets will grow to define our industry
- Secure and reliable interoperability of data, chains, and identity are needed to make transactions work
- Compliance through clear regulation is the last critical unlock for capital markets adoption

Real-World Asset Tokenization is the Future of Our Industry



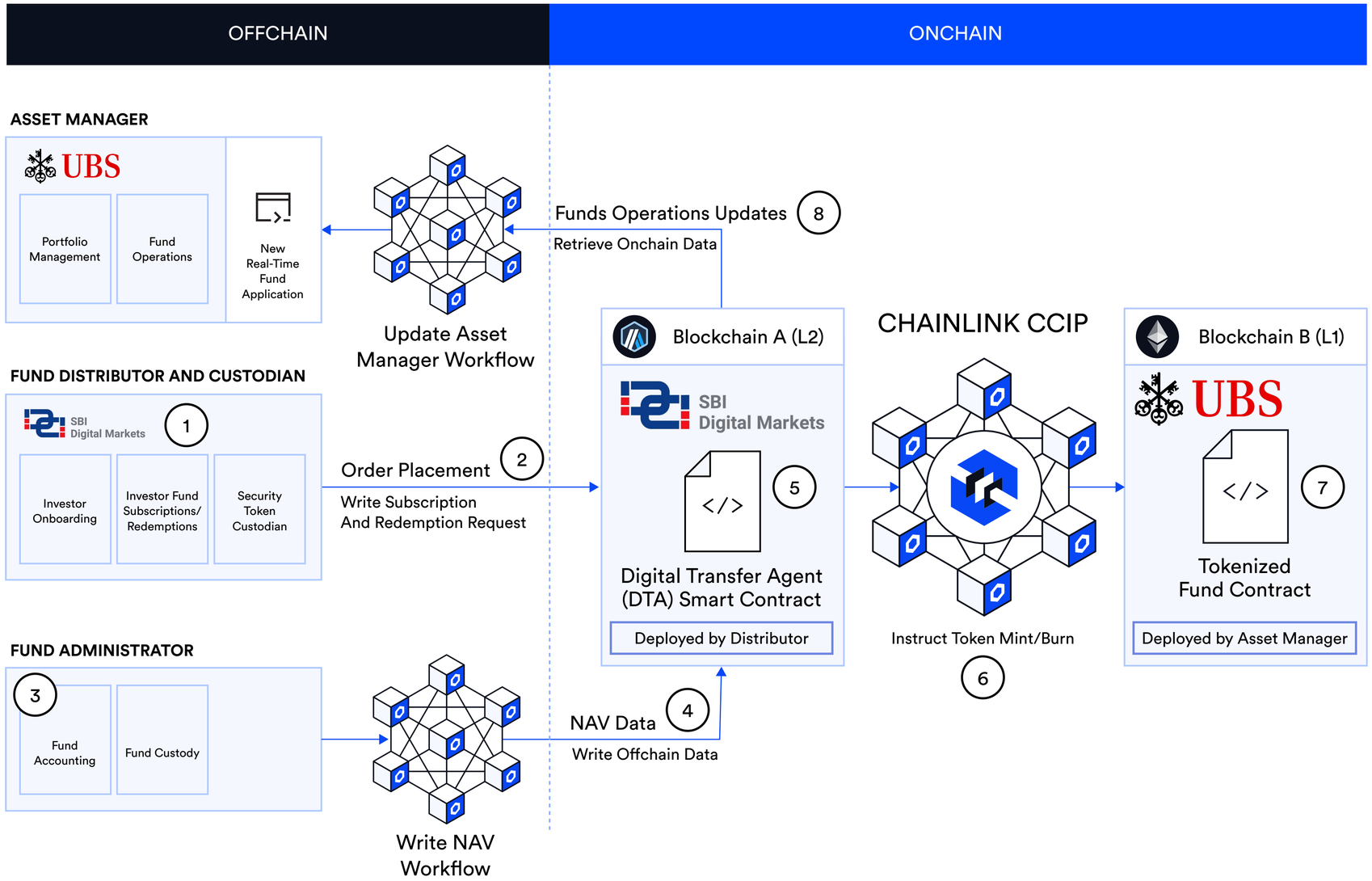
The Market for Onchain Assets Is Hundreds of Trillions



Compliant Capital is the Largest Driver of Future Growth

- Non-compliant retail capital can grow our industry to $10 trillion+ through retail capital flows
- Compliant capital/institutions can grow our industry to $100 trillion+ through institutional capital flows
- We need to solve how how compliant payments, compliant assets, and compliant transactions work

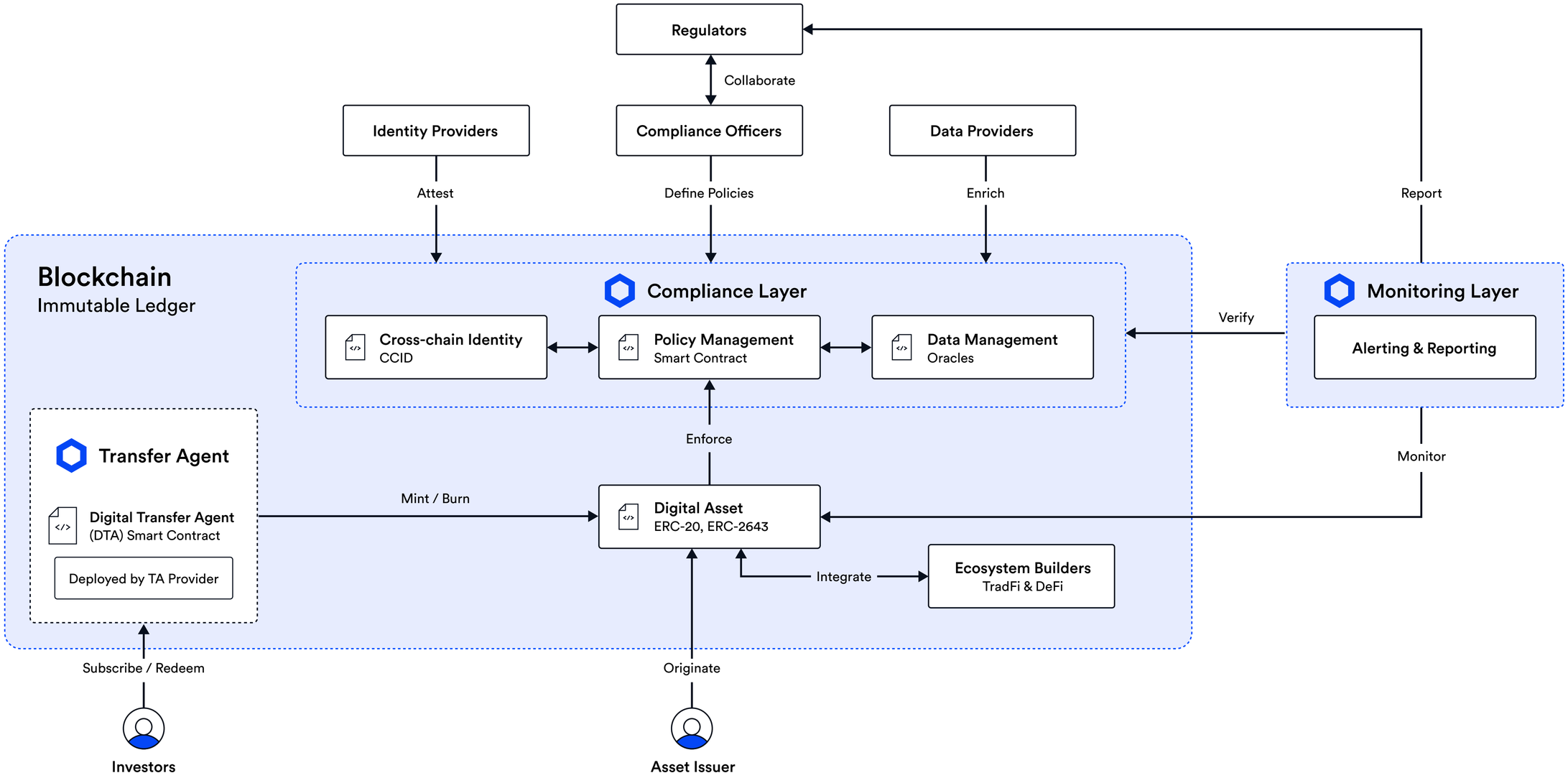
Enabling Next Generation Tokenized Funds for UBS and SBI



1. Investors submit subscription or redemption orders for UBS's tokenized fund through SBI Digital
2. Orders are sent to the Digital Transfer Agent Smart Contract and held in pending status
3. Fund Administrator calculates and validates the fund's NAV for the dealing cycle
4. Chainlink securely delivers the validated NAV data to the Digital Transfer Agent Smart Contract
5. Digital Transfer Agent (DTA) smart contract calculates required token amounts based on NAV and updates order status inside of smart contract
6. Digital Transfer Agent (DTA) uses Chainlink CCIP to trigger validated minting/burning by the tokenized fund contract, possibly across various chains
7. Fund Contract mint/burns tokens and SBI Digital Market allocates shares to investors in backend
8. Throughout the entire process, UBS maintains complete visibility of all fund operations through Chainlink's oracle network, which provides real-time data flows between their traditional systems and the blockchain-based Digital Transfer Agent - enabling seamless integration of this new tokenized fund service into their existing infrastructure

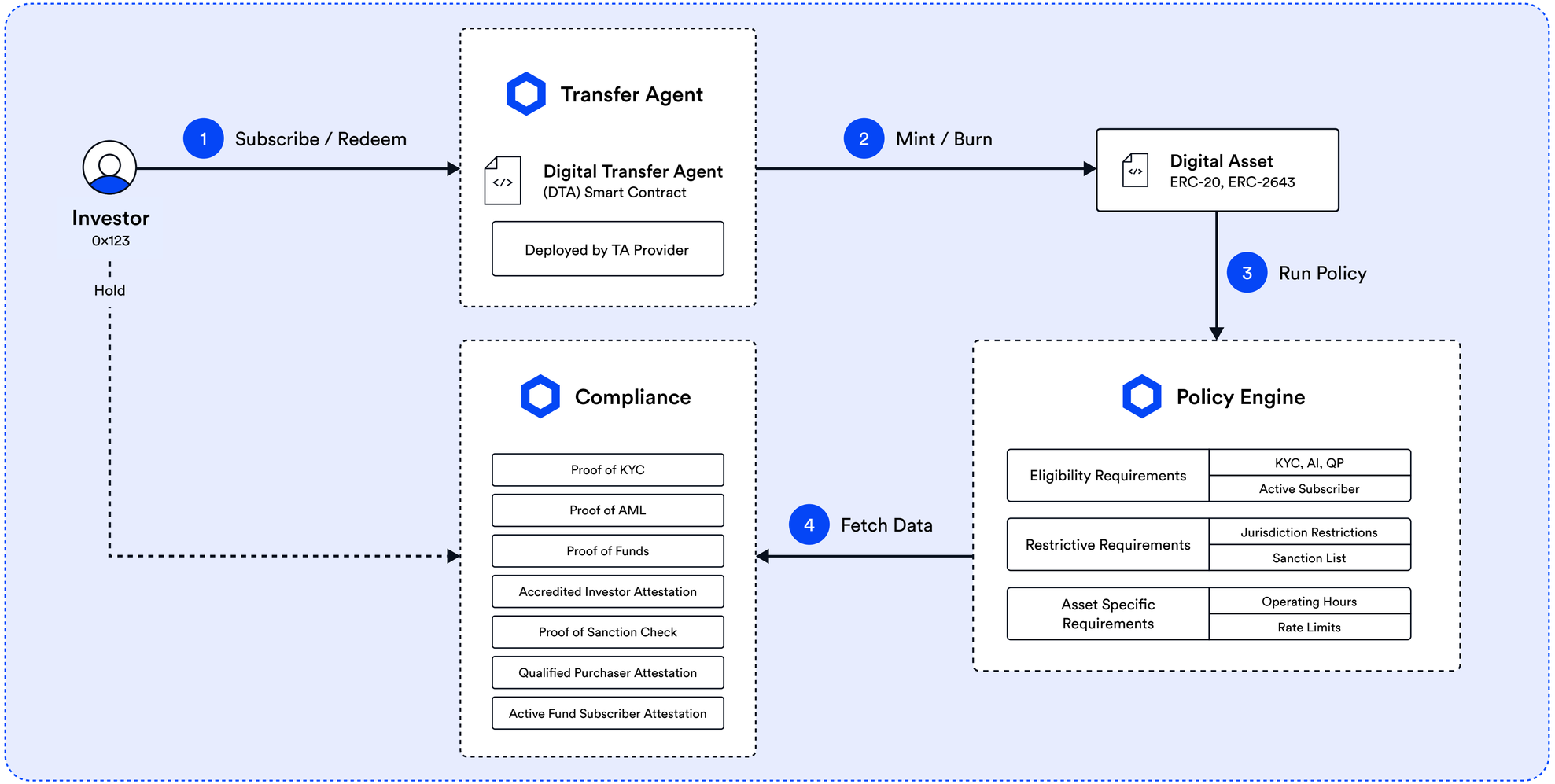
Automated Policy Enforcement

Ensuring onchain operations adhere to compliance requirements.



Policy Management Engine

Facilitates rules definition and automated enforcement, resulting in accelerated external investor onboarding onchain.



Many Digital Assets Bought in Various Forms of Payment

- Assets are necessary to grow the inventory of what can be bought/transacted against via payments
- Payment solutions are needed to provide access to frictionless purchasing of assets by users/liquidity
- Compliant assets and payments interacting with each other onchain requires lots of orchestration

Chainlink Runtime Environment (CRE)

A secure, compliance-focused, reliable infrastructure for building and running financial applications.

Abstracts Blockchain Complexity

Dramatically Accelerates Web3 Applications Development and Integration

Provides enterprise-Grade Orchestration of financial workflows across Web2 and Web3 platforms

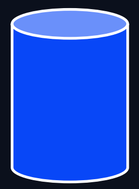
Enables financial institutions to integrate new capabilities and workflows without disrupting legacy infrastructure and processes

Modularity by design

CRE Components and Capabilities

EXAMPLES

Capabilities



- CRON Trigger
- Chain Reader
- Compute
- Chain Writer
- etc…

Can interact to any chain CRE is integrated with

Adapters

- Swift/ISO
- FIX
- DTCC
- etc…

Foundational Smart Contracts

- DvP Coordinator
- DTA Manager
- Feed Proxy
- etc..

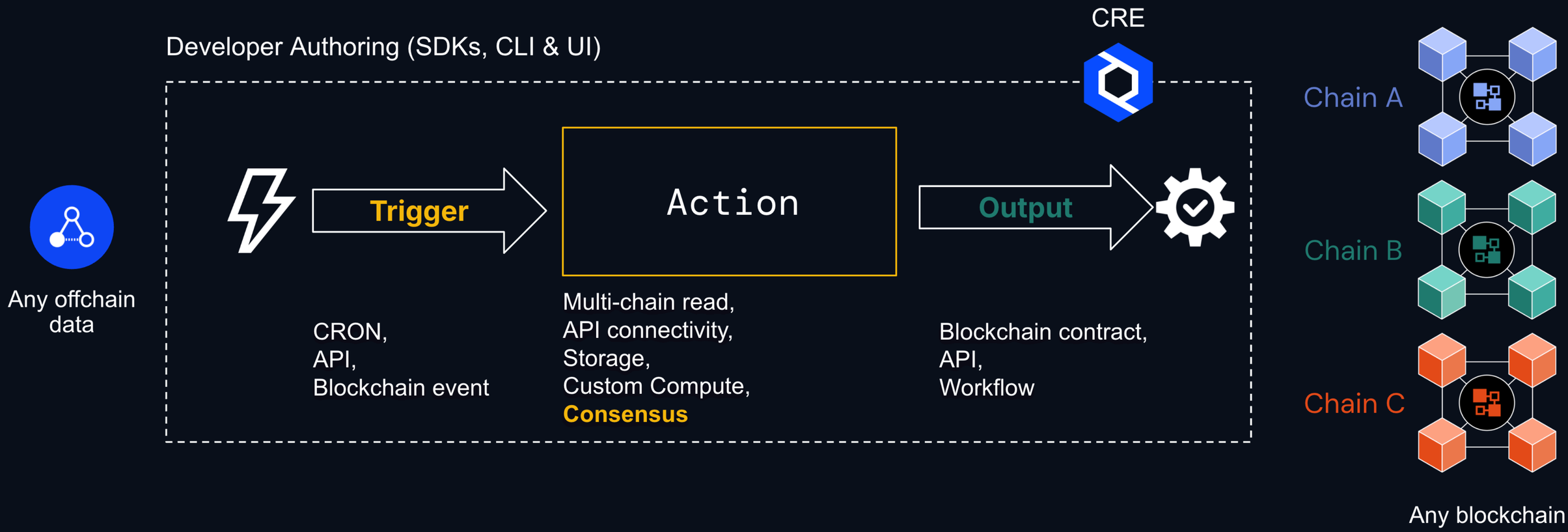
Business Workflows

- Blueprints
- Workflow Engine
- (Pre-Built) Composable Business Workflows

How to use CRE

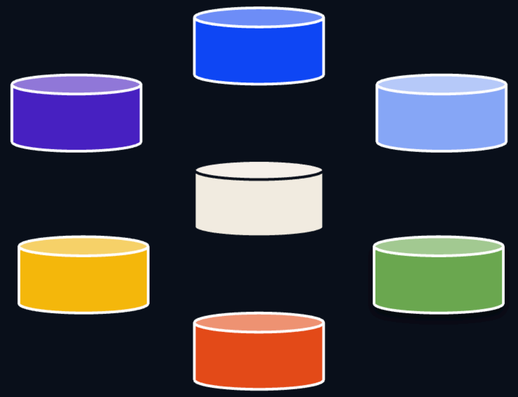
Use Capability SDKs to write Workflows

Seamless Programmability and Scalability Across All Blockchains

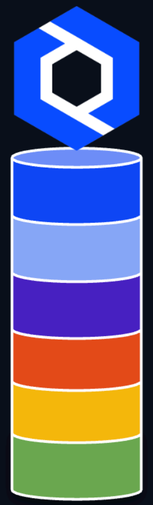


Custom “Workflows” address unique challenge

Chainlink Capabilities

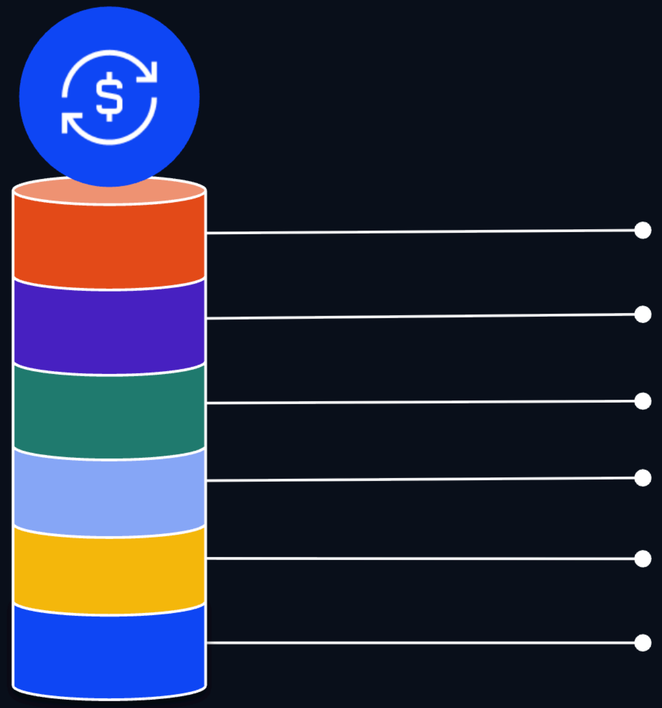


Developer Workflow



CRE

Example: Cross-Chain Liquidity Balancer



- CRON Trigger
- Chain Reader: Read balances from multiple chains
- API Fetch: Data on relative demand for tokens
- Compute: Pools to increase, pools to decrease, tokens to swap
- Consensus
- Chain Writer: Instructions and reports

Example workflow in GO

CRON Trigger

Compute

API Fetch

Multi-variable consensus

Chain Writer

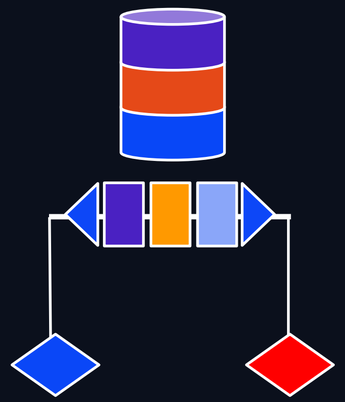
```go
func BuildWorkflow(config []byte) *sdk.WorkflowSpecFactory {
    workflow := sdk.NewWorkflowSpecFactory()

    cron := croncap.Config{
        Schedule: "*/30 * * * * *", // Every 30 seconds
    }.New(workflow)

    compConf := computeConfig{
        FeedID: "[redacted]",
    }

    compute := sdk.Compute1WithConfig(
        workflow,
        "compute",
        &sdk.ComputeConfig[computeConfig]{Config: compConf},
        sdk.Compute1Inputs[croncap.Payload]{Arg0: cron},
        func(runtime sdk.Runtime, config computeConfig, outputs croncap.Payload) (computeOutput, error) {
            feedID, err := convertFeedIDtoBytes(config.FeedID)
            if err != nil {
                return computeOutput{}, fmt.Errorf("cannot convert feedID to bytes : %w : %b", err, feedID)
            }

            fresp, err := runtime.Fetch(sdk.FetchRequest{
                URL:       "[redacted]",
                Method:    "GET",
                TimeoutMs: 5000,
            })
            if err != nil {
                return computeOutput{}, err
            }

            var resp trueUSDResponse
            err = json.Unmarshal(fresp.Body, &resp)
            if err != nil {
                return computeOutput{}, err
            }

            if resp.Ripcord {
                err := runtime.Emitter().With(
                    "feedID", config.FeedID,
                ).Emit(fmt.Sprintf("ripcord flag set for feed ID %s", config.FeedID))
                if err != nil {
                    runtime.Logger().Error("failed to emit custom message")
                }
                return computeOutput{}, sdk.BreakErr
            }

            return computeOutput{
                Price: int(resp.TotalTrust * 100),
                FeedID:    feedID, // TrueUSD
                Timestamp: resp.UpdatedAt.Unix(),
            }, nil
        },
    )

    consensusInput := ocr3cap.ReduceConsensusInput[computeOutput]{
        Observation: compute.Value(),
    }

    consensus := ocr3cap.ReduceConsensusConfig[computeOutput]{
        Encoder: ocr3cap.EncoderEVM,
        EncoderConfig: map[string]any{
            "abi": "(bytes32 FeedID, uint32 Timestamp, bytes Bundle)[] Reports",
            "subabi": map[string]string{
                "Reports.Bundle": "uint224 Price",
            },
        },
        ReportID: "0001",
        KeyID:    "evm",
        AggregationConfig: aggregators.ReduceAggConfig{
            Fields: []aggregators.AggregationField{
                {
                    InputKey:  "FeedID",
                    OutputKey: "FeedID",
                    Method:    "mode",
                },
                {
                    InputKey:        "Timestamp",
                    OutputKey:       "Timestamp",
                    Method:          "median",
                    DeviationString: "300",
                    DeviationType:   "absolute",
                },
                {
                    InputKey:        "Price",
                    OutputKey:       "Price",
                    Method:          "median",
                    DeviationString: "1",
                    DeviationType:   "percent",
                    SubMapField:       true,
                },
            },
            ReportFormat: aggregators.REPORT_FORMAT_ARRAY,
            SubMapKey: "Bundle",
        },
    }.New(workflow, "consensus", consensusInput)

    targetInput := chainwriter.TargetInput{
        SignedReport: consensus,
    }

    chainwriter.TargetConfig{
        Address:    "[redacted]", // DF2.0 Cache
        DeltaStage: "15s",
        Schedule:   "oneAtATime",
    }.New(workflow, "write_ethereum-testnet-sepolia@1.0.0", targetInput)

    return workflow
}
```

Introducing CRE

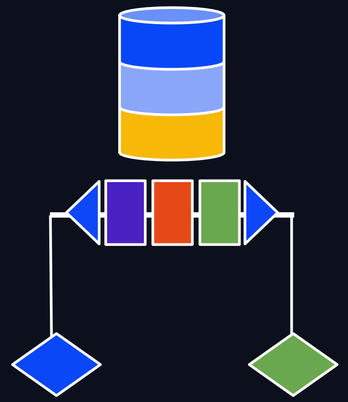
Customize pre-built workflows or build your own in a self-serve manner

Build in weeks, not months

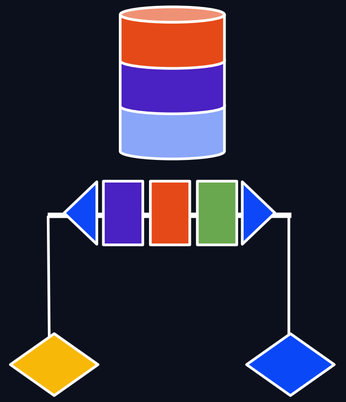
Digital Transfer Agent



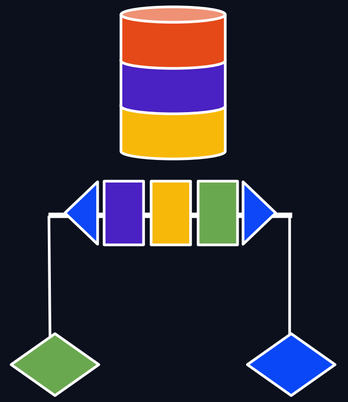
Delivery Versus Payment



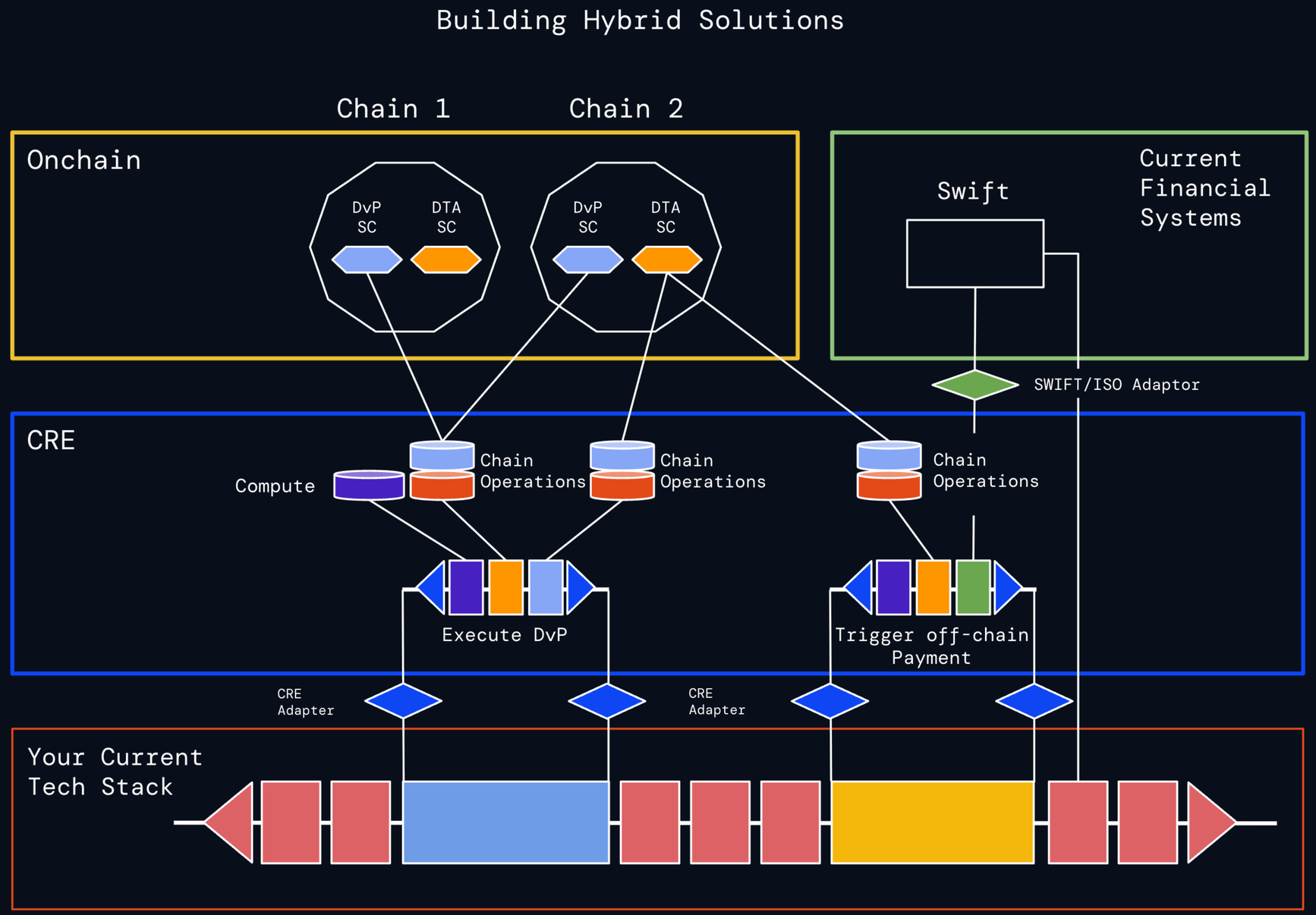
Payment Processor



Cross-Chain Token Management



Integrate them with your current workflows and systems

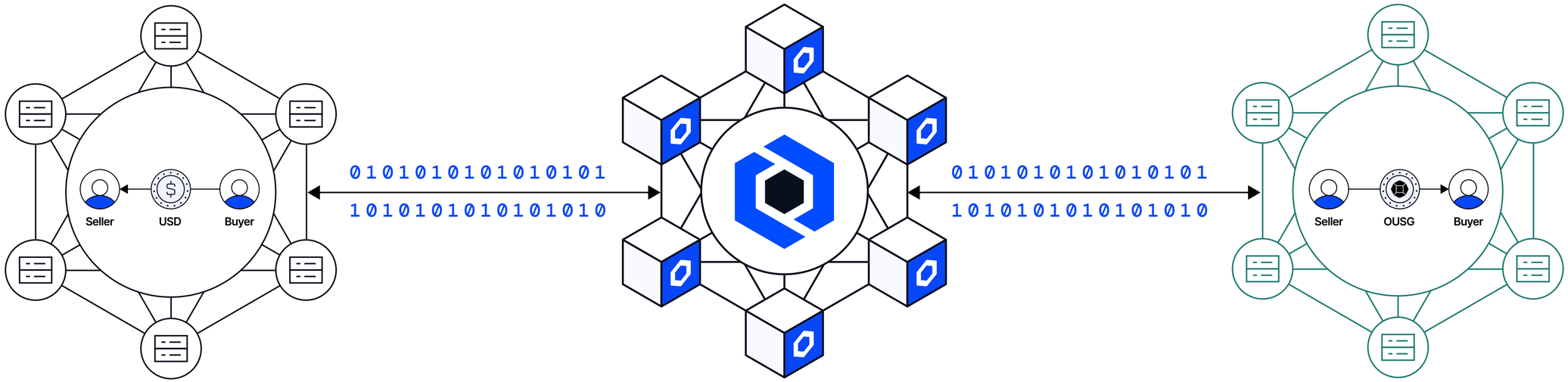


Kinexys Private Chain Paying for Ondo Public Chain Assets

kinexys by J.P.Morgan

Chainlink

Ondo Chain



Private Blockchain

Chainlink Runtime Environment

RWA Blockchain

Chainlink

Thank You

Disclaimer: This presentation is for informational purposes only and contains statements about the future, including anticipated programs and features, developments, and timelines for the rollout of these programs and features. These statements are only predictions and reflect current beliefs and expectations with respect to future events; they are based on assumptions and are subject to risk, uncertainties, and change at any time. There can be no guarantee that any of the contemplated programs or features will be implemented as specified nor any assurance that actual results will not differ materially from those expressed in these statements, although we believe them to be based on reasonable assumptions. All statements are valid only as of the date first presented. The statements in this presentation also may not reflect future developments due to user feedback or later events and we may not update this presentation in response.