SOLIDO

Technical Introduction Paper

SAFV – Supra AutoVault Framework

January 2nd, 2026

Authors: Solido Money

Email: contact@solido.money

Abstract

Supra introduces AutoVaults as a native AutoFi primitive, enabling automated and trustless execution of financial strategies through MoveVM, native automation, and oracle-first infrastructure. These primitives provide the execution guarantees required for autonomous on-chain finance. As AutoVault-based applications evolve, standardized construction patterns, consistent interfaces, and reusable abstractions become essential to ensure composability, auditability, and developer efficiency.

This paper introduces SAFV (Supra AutoVault Framework), a middleware framework that standardizes the construction of strategy-driven vaults and the issuance of strategy-backed on-chain assets on top of Supra's AutoFi stack. Within SAFV, an asset is defined as a tokenized claim on a vault, where vault behavior is governed by deterministic, pre-defined strategy logic. Vaults act as capital containers, strategies encode execution rules, and vault tokens represent proportional ownership of managed assets.

SAFV draws from established DeFi vault standards and structured finance principles to support a broad class of automated strategies, including index allocation, delta-neutral yield, and rule-based trading systems. By combining DeFi composability, systematic strategy design, and Supra's native execution guarantees, SAFV enables transparent, scalable, and programmatically enforced on-chain financial assets. The framework is designed to address immediate DeFi use cases while remaining extensible toward future strategy composition paradigms, including AI-driven signal execution and no-code strategy construction.

SOLIDO

Contents

1. Background and Motivation

1.1 Supra AutoFi and the Emergence of AutoVaults

Supra's AutoFi stack introduces a vertically integrated execution environment for autonomous on-chain finance. By combining MoveVM's resource-oriented execution model with native automation and oracle-first infrastructure, Supra enables financial logic to be executed deterministically, transparently, and without reliance on off-chain keepers or discretionary operators.

Within this stack, **AutoVaults** serve as execution containers capable of holding assets and performing automated actions according to predefined rules. AutoVaults define what is possible at the execution layer: trustless automation, bounded execution, and real-time data-driven behavior.

SAFV builds directly on these guarantees. It does not alter or extend Supra's execution primitives; rather, it focuses on how AutoVaults are constructed, parameterized, and composed into reusable financial products.

1.2 Terminology: Strategy-Backed Assets

Throughout this paper, the term "asset" refers specifically to a tokenized claim on a strategy-driven vault.

An asset in the SAFV context is not:
- a standalone speculative token,
- a discretionary financial instrument,
- or a governance representation.

Instead, it represents proportional ownership in a vault whose behavior is fully defined by deterministic strategy logic enforced on-chain. Vault tokens issued by SAFV inherit their economic behavior exclusively from the underlying vault and its associated strategy.

This definition aligns asset value directly with transparent execution rather than discretionary control.

1.3 DeFi as the Immediate Design Environment

While structured finance concepts inform the long-term vision of strategy-backed assets, the immediate environment for SAFV is DeFi-native strategy execution.

Modern DeFi already supports a wide range of systematic strategies, including:
- delta-neutral yield farming,
- automated liquidity provisioning,
- funding-rate arbitrage,
- index-style asset exposure,
- rule-based trading systems.

These strategies collectively manage significant on-chain capital but are often implemented using:

- bespoke smart contracts,
- off-chain automation,
- or centralized execution infrastructure.

AutoVaults eliminate these dependencies by enabling native, trustless execution. SAFV further lowers the barrier to entry by providing a standardized construction framework that allows strategists to focus on strategy design rather than infrastructure engineering.

1.4 The Early Standardization Opportunity

At the current stage of AutoFi adoption, advanced AutoVault implementations remain limited. This creates an opportunity to establish shared abstractions and interfaces **before divergent patterns emerge.**

Early standardization offers several advantages:

- improved auditability through familiar interfaces,
- reduced cognitive load for developers,
- predictable behavior across vault products,
- and stronger composability across protocols.

SAFV is intentionally designed as an early construction framework, providing canonical patterns for vault lifecycle management, strategy integration, and asset issuance as AutoVault adoption accelerates.

1.5 From Execution Primitives to Financial Products

Supra AutoFi defines the execution foundation for autonomous finance. SAFV bridges the gap between that foundation and production-grade financial products by introducing a consistent framework for building strategy-driven vaults and issuing strategy-backed assets.

By separating execution guarantees from construction standards, SAFV enables:

- rapid experimentation without sacrificing safety,
- consistent product behavior across implementations,
- and a scalable path toward more expressive strategy composition in the future.

2. Conceptual Foundations

This section formalizes the core concepts used throughout SAFV. These concepts are intentionally simple in isolation but powerful in composition. SAFV does not introduce new financial primitives; instead, it provides a structured way to **compose existing primitives into deterministic, automated financial products.**

2.1 Signals

A signal is any input that informs strategy behavior.

Signals do not execute actions and do not control capital. They provide information that can be evaluated by strategy logic. In SAFV, signals are treated as read-only inputs and may originate from multiple sources, including:

- On-chain oracle feeds (e.g., price, volatility, funding rates)
- Time-based triggers
- Signed inputs from humans, DAOs, or automated agents
- Algorithmic or AI-generated outputs recorded on-chain

By design, signals are decoupled from execution authority. No signal source—human, algorithmic, or automated—can directly move funds.

SAFV does not define or constrain how signals are produced; it only specifies how signals are consumed by strategies within the framework.

2.2 Strategies

A strategy is a deterministic set of rules that consumes signals and proposes actions.
Strategies define:

- how signals are interpreted,
- under what conditions actions are taken,
- and which actions are permissible.

Examples include:
- rebalancing asset weights when thresholds are crossed,
- rotating exposure based on momentum criteria,
- hedging positions when risk bounds are exceeded.

Strategies in SAFV are bounded by construction. They cannot:
- withdraw assets arbitrarily,
- bypass vault accounting,
- or violate predefined constraints.

This mirrors mandate-based execution in traditional finance, where operators act within strict, pre-defined rules rather than discretionary control.

2.3 Vaults

A **vault** is a capital container that:
- holds user-deposited assets,
- executes strategy-approved actions,
- and issues vault tokens representing ownership.

Vaults are responsible for:

- custody and accounting,
- enforcing entry and exit rules,
- applying fees and delays,
- and maintaining invariant guarantees.

In SAFV, vaults are intentionally strategy-agnostic. They do not embed financial logic directly; instead, they expose standardized interfaces that strategies may interact with under strict constraints.

2.4 Vault Tokens

Vault tokens represent proportional ownership in a strategy-driven vault.

Key properties include:

- ownership proportional to contributed capital,
- value derived from vault net asset value,
- behavior fully determined by the underlying strategy.

Vault tokens are not governance instruments and do not grant control rights. They function analogously to units of a systematic fund, with transparent on-chain accounting and deterministic behavior.

2.5 Composition Model

SAFV formalizes the following composition:

Layer	Role
Signals	Provide information
Strategies	Interpret signals and propose actions
Vaults	Enforce rules and execute actions
Vault Tokens	Represent ownership and exposure

This separation of concerns enables modular strategy design, safer execution, easier auditing, and long-term composability.

2.6 DeFi and Structured Finance Parallels

While SAFV is designed primarily for DeFi-native strategies, its composition model aligns closely with structured finance concepts:

TradFi Concept	SAFV Equivalent
Market data	Signals
Investment mandate	Strategy rules
Fund vehicle	Vault
Fund units	Vault tokens

This alignment allows DeFi strategies to be expressed with the same structural clarity as traditional systematic products, while retaining the benefits of on-chain execution.

3. SAFV Architecture and Framework Design

SAFV is designed as a middleware construction framework that standardizes how strategy-driven vaults are built on top of Supra's AutoFi execution primitives. It does not introduce new execution guarantees; instead, it defines consistent abstractions, interfaces, and lifecycle rules that enable safe, composable, and auditable AutoVault-based products.

At a high level, SAFV decomposes vault construction into four core components:

1. Vault Core
2. Strategy Interface
3. Execution & Automation Hooks
4. Asset Issuance Layer

Each component has a clearly defined responsibility and explicit boundaries.

3.1 Architectural Positioning

SAFV sits strictly above Supra AutoFi primitives and below end-user applications.

Layer	Description
Supra AutoFi	MoveVM execution, native automation, oracles, risk primitives
SAFV	Vault construction framework, standards, lifecycle management
Applications	Index vaults, delta-neutral vaults, trading and yield strategies

This separation ensures that:
- execution guarantees remain a protocol-level concern,
- construction standards evolve independently,
- applications remain composable and interoperable.

3.2 Vault Core

The **Vault Core** is the foundational component of SAFV. It defines a standardized vault interface responsible for capital custody, accounting, and invariant enforcement.

Responsibilities
- Accepting and managing user deposits
- Issuing and burning vault tokens
- Enforcing withdrawal rules and delays
- Applying fee logic
- Maintaining deterministic accounting

The Vault Core is strategy-agnostic. It does not embed financial logic or assumptions about asset allocation. All strategy behavior is expressed externally and executed through constrained interfaces.

This design is inspired by ERC-4626–style vault abstractions, adapted to Supra's MoveVM and AutoFi environment.

3.3 Vault Lifecycle

SAFV formalizes a consistent vault lifecycle to reduce implementation variance and audit complexity.

Lifecycle Phases
1. Initialization
2. Deposit / Mint
3. Strategy Execution

4. Rebalancing / Adjustment
5. Withdrawal / Redemption
6. Shutdown (optional)

Phase	Description
Initialization	Vault parameters, asset type, and constraints are fixed
Deposit	Users contribute assets and receive vault tokens
Execution	Strategies propose actions based on signals
Rebalancing	Asset allocation is adjusted within bounds
Withdrawal	Assets are redeemed under enforced rules
Shutdown	Vault ceases operation under predefined conditions

All lifecycle transitions are deterministic and rule-based.

3.4 Strategy Interface

Strategies interact with vaults through a **standardized strategy interface** defined by SAFV.

Strategy Capabilities

Strategies may:
- read vault state,
- evaluate signals,
- propose bounded actions (e.g., swap, rebalance, hedge).

Strategies may not:
- withdraw funds directly,
- override vault accounting,
- bypass safety constraints.

This interface ensures that strategies remain **pure logic modules,** while vaults retain full control over capital.

3.5 Execution and Automation Hooks

Execution is delegated to Supra's native automation layer. SAFV defines how strategies express intent, not how execution is scheduled or triggered.

Supported execution models include:
- time-based execution,
- oracle-threshold execution,
- signal-triggered execution.

By separating intent from execution, SAFV ensures that:
- automation remains trustless,
- strategies remain portable,
- execution guarantees remain consistent.

3.6 Asset Issuance Layer (Vault Tokens)

SAFV standardizes the issuance of vault tokens as the canonical representation of strategy-backed assets.

Properties of Vault Tokens
- Represent proportional ownership of vault assets
- Reflect net asset value deterministically
- Are minted and burned exclusively by the Vault Core
- Do not convey governance or discretionary control

Vault tokens are intentionally minimal and composable, enabling integration with broader DeFi ecosystems without embedding protocol-specific logic.

3.7 Safety and Constraint Model

SAFV enforces a layered constraint model to eliminate common failure modes.

Constraint Type	Purpose
Accounting constraints	Prevent balance inconsistencies
Strategy bounds	Limit permissible actions
Withdrawal rules	Enforce delays and limits

Constraint Type	Purpose
TVL caps	Control capacity and risk
Emergency controls	Enable circuit-breaking

All constraints are enforced programmatically and cannot be bypassed by strategy logic.

3.8 Design Principles

SAFV adheres to the following design principles:

- **Separation of concerns**
  Execution, construction, and application layers remain distinct.
- **Determinism over discretion**
  All behavior is rule-based and transparent.
- **Composability by default**
  Interfaces are standardized and minimal.
- **Auditability at scale**
  Familiar patterns reduce review overhead.

These principles ensure SAFV can scale alongside the Supra ecosystem without introducing systemic complexity.

4. Strategy Archetypes

SAFV is designed to support a broad range of automated strategies while maintaining a consistent construction and execution model. Rather than prescribing specific financial products, the framework defines how strategies are expressed and enforced, enabling diverse use cases to be built on a common foundation.

This section outlines key strategy archetypes that SAFV supports today, with an emphasis on DeFi-native strategies that already manage significant on-chain capital.

4.1 Index and Basket Strategies

Index strategies allocate capital across a predefined basket of assets according to fixed or rule-based weights. In SAFV, index strategies are implemented by combining oracle-driven signals with deterministic rebalancing logic.

Typical characteristics include:
- predefined asset sets and weights,
- periodic or threshold-based rebalancing,
- transparent accounting and valuation.

Index vaults built using SAFV issue vault tokens that represent proportional exposure to the underlying basket. Rebalancing actions are executed automatically within predefined bounds, ensuring predictable behavior and minimizing discretionary intervention.

These strategies mirror both DeFi index products and traditional index funds, while benefiting from on-chain transparency and automated enforcement.

4.2 Delta-Neutral Yield Strategies

Delta-neutral strategies aim to generate yield while minimizing directional exposure to underlying assets. In DeFi, such strategies often combine yield-bearing positions with hedging mechanisms to neutralize market risk.

Within SAFV, delta-neutral strategies are expressed as:
- bounded allocation rules,
- exposure constraints enforced by the vault,
- automated rebalancing in response to changing market conditions.

By encoding these constraints directly into strategy logic and vault rules, SAFV enables delta-neutral strategies to operate autonomously without reliance on off-chain execution or discretionary management.

4.3 Rule-Based Trading and Momentum Strategies

Rule-based trading strategies use deterministic logic to adjust exposure based on market signals. Common examples include momentum strategies, trend-following systems, and trailing stop-loss mechanisms.

In SAFV:
- strategies evaluate price and volatility signals,
- actions are proposed only when predefined conditions are met,
- vaults enforce execution boundaries and capital safety.

This approach enables transparent, auditable trading strategies where all behavior is encoded in advance and executed trustlessly on-chain.

4.4 Automated Liquidity Management

Liquidity provisioning strategies aim to optimize returns from providing liquidity to decentralized exchanges while managing associated risks.

SAFV supports automated liquidity management by:
- defining strategy logic for position entry and exit,
- responding to oracle-based market signals,
- enforcing exposure and risk constraints at the vault level.

These strategies can be deployed without bespoke contract development, reducing implementation risk and improving consistency across deployments.

4.5 Strategy Parameterization and Reuse

A key design goal of SAFV is to enable **strategy reuse through parameterization.**

Rather than deploying entirely new contracts for each strategy variant, strategists can:
- reuse core strategy templates,
- adjust parameters such as asset sets, thresholds, or timing,
- deploy multiple vaults with predictable behavior.

This model improves audit efficiency and accelerates experimentation while maintaining safety guarantees.

4.6 Extensibility Toward Advanced Strategy Classes

While SAFV prioritizes immediate DeFi use cases, the framework is intentionally extensible.

Advanced strategy classes may include:
- multi-strategy compositions,
- adaptive parameter tuning,
- AI-generated or externally signed signal inputs.

In all cases, execution remains bounded by vault constraints, ensuring that increased strategy sophistication does not compromise safety or determinism.

5. Vault Token Composability and Use Cases

Vault tokens issued under the SAFV framework are designed to function as first-class on-chain assets, rather than as isolated receipts or protocol-specific instruments.

Because vault tokens represent proportional ownership of assets managed under deterministic, rule-based strategies, they inherit several properties that enable broader composability across decentralized financial systems.

5.1 Use as Collateral

A primary use case for SAFV vault tokens is their use as collateral within other on-chain financial protocols.

Vault tokens:
- have transparent and deterministic valuation derived from vault net asset value
- operate under immutable or bounded strategy logic
- do not embed discretionary withdrawal or execution authority

These properties make vault tokens suitable for integration into lending markets, margin systems, and structured borrowing products, enabling users to unlock liquidity while maintaining exposure to underlying strategies.

5.2 Layered and Composed Products

Because SAFV standardizes vault behavior and asset issuance, vault tokens may themselves be composed into higher-order financial products.
Examples include:
- index constructions composed of multiple vault tokens
- meta-vaults that allocate capital across underlying strategies
- products that rebalance exposure between vault tokens based on predefined rules

SAFV does not prescribe these constructions; it ensures that vault tokens behave predictably when used as building blocks.

5.3 Support for More Expressive Strategies

Standardized vault construction lowers the marginal cost of deploying and auditing more sophisticated strategies.
By separating custody, execution, and strategy logic, SAFV enables:
- multi-leg and conditional strategies
- dynamic allocation across strategy states
- strategy compositions that adapt across market regimes

All such strategies remain subject to vault-level constraints and deterministic execution guarantees.

5.4 Beyond Yield-Centric Designs

While many early vault strategies focus on yield generation, SAFV vault tokens are not limited to yield-only use cases.

The framework supports the construction of:
- index-style exposure products
- volatility-managed or risk-bounded strategies
- capital-constrained or mandate-driven constructions

SAFV's role is to provide a consistent asset substrate; financial product design remains an application-layer concern.

6. Security, Risk Controls, and Unruggability

Security in SAFV is achieved through structural constraints, not through trust assumptions or discretionary controls. The framework is designed such that no single actor—strategy author, signal provider, or automation trigger—can unilaterally extract funds or violate predefined rules.

Rather than treating security as an afterthought, SAFV embeds risk controls directly into vault construction and lifecycle management.

6.1 Separation of Authority

SAFV enforces a strict separation between:
- information providers (signals),
- decision logic (strategies),
- and capital custody (vaults).

Signals provide data but have no execution authority.
Strategies propose actions but cannot directly move funds.
Vaults enforce all accounting, constraints, and asset transfers.

This separation eliminates a common class of DeFi exploits where control over logic implicitly grants control over funds.

6.2 Deterministic and Bounded Execution

All vault behavior in SAFV is:
- deterministic,
- rule-based,
- and bounded by construction.

Strategies operate within explicitly defined limits:

- permitted asset types,
- maximum position sizes,
- allowable execution paths.

No strategy can introduce new behavior after deployment, override constraints, or bypass vault logic. This mirrors mandate-based controls in traditional systematic funds, where execution is limited by predefined investment rules.

6.3 Withdrawal Controls and Capital Safety

SAFV supports multiple mechanisms to protect capital during entry and exit:

- Withdrawal delays to mitigate rapid liquidity shocks
- Minimum withdrawal thresholds to prevent dust or griefing attacks
- TVL caps to manage capacity and concentration risk

These controls are enforced programmatically and apply uniformly to all users, ensuring predictable and fair treatment.

6.4 Emergency Controls and Circuit Breakers

SAFV vaults may include emergency controls designed to respond to unexpected conditions, such as:

- oracle anomalies,
- extreme market volatility,
- or infrastructure failures.

Emergency actions are limited in scope and do not grant discretionary withdrawal rights. Their purpose is to pause execution or restrict new actions, not to enable fund extraction.

This design aligns with circuit breaker mechanisms commonly used in traditional financial systems, translated into deterministic on-chain logic.

6.5 Elimination of Common Rug Vectors

SAFV materially reduces common rug-pull vectors by design:

Common Risk	SAFV Mitigation
Admin-controlled withdrawals	No unilateral withdrawal authority

Common Risk	SAFV Mitigation
Hidden strategy changes	Immutable strategy logic
Off-chain execution	Native on-chain automation
Opaque accounting	Deterministic vault accounting
Privileged access	Uniform rules for all participants

Unruggability in SAFV is not a marketing claim; it is a consequence of constrained system design.

6.6 Auditability and Review Efficiency

By standardizing vault interfaces and strategy boundaries, SAFV improves audit efficiency:

- auditors review familiar patterns,
- strategy logic is isolated from custody logic,
- repeated deployments reuse known abstractions.

This reduces both the cost and complexity of security reviews as the ecosystem scales.

6.7 Risk Is Explicit, Not Eliminated

SAFV does not claim to eliminate financial risk. Strategy performance remains subject to:

- market conditions,
- signal quality,
- and parameter selection.

What SAFV does eliminate is structural risk arising from discretionary control, opaque execution, or inconsistent vault behavior. All remaining risk is explicit, transparent, and attributable to strategy design.

7. Case Study — Solido Grow

Solido Grow is a production deployment that demonstrates how the abstractions defined in SAFV operate in practice. The system applies SAFV's vault, strategy, and execution model to a live yield-generating product on Supra, validating the framework under real market conditions.

7.1 Vault Architecture and Lifecycle

Solido Grow is implemented as a single vault contract that supports multiple strategy paths, segmented by collateral type.

- A unified vault manages custody and accounting for $CASH deposits
- Distinct strategies are applied per collateral class (e.g., SUPRA, stSUPRA)
- The architecture is designed to scale horizontally to additional collateral types (e.g., iBTC, iETH) without introducing new vault primitives

Vault parameters such as fees, withdrawal delays, and risk thresholds are governance-configurable, allowing the system to adapt over time while preserving standardized construction.

This design directly reflects SAFV's Vault Core abstraction: strategy-agnostic custody with parameterized behavior.

7.2 Strategy Implementation

Each Solido Grow strategy is implemented as bounded logic operating against a specific collateral type. In practice:

- strategies monitor oracle prices and collateralization ratios
- under-collateralized positions in the Solido Cash protocol are liquidated according to deterministic rules
- a portion of the captured value is routed into the Grow vault

Strategies cannot:

- withdraw funds arbitrarily
- bypass vault accounting
- alter execution paths at runtime

This separation of custody (vault) and logic (strategy) is a direct application of SAFV's strategy interface model.

7.3 AutoFi Execution Model

All strategy execution in Solido Grow is handled by Supra's native AutoFi automation.

- AutoFi bots are currently whitelisted, prioritizing safety during early operation
- A single execution path is used today, with redundancy considered a future enhancement
- Execution is fully on-chain and triggered by protocol state and oracle conditions

No off-chain keepers or discretionary operators are involved. This validates SAFV's assumption that execution intent and scheduling can be cleanly separated from strategy logic.

7.4 Safety Controls and Risk Management

Solido Grow includes multiple layers of safety controls:

- strategy-level bounds on execution
- vault-level accounting invariants
- automatic execution halts under abnormal conditions (e.g., excessive slippage or detected malicious behavior)

Emergency protections are automatically enforced and do not grant discretionary withdrawal authority. To date, no emergency pause has been triggered in production.

These controls directly informed SAFV's unruggability and constraint model.

7.5 System Outcomes

Solido Grow has delivered 40%+ APY on $CASH deposits consistently since inception, generated entirely through on-chain, rule-based execution.

Beyond yield, the system has produced meaningful protocol-level effects:

- liquidations are absorbed smoothly by the Grow vault
- higher sustainable LTVs are supported in Solido Cash
- capital efficiency improves without increasing systemic risk

During market downturns, the strategy exhibits counter-cyclical behavior, accumulating $CASH and reinforcing peg stability through structural buying pressure.

7.6 Asset Accounting Snapshot

Yield accrues through exchange-rate appreciation between $bCASH and $CASH rather than explicit distribution.

> *Illustrative snapshot (non-binding): at the time of writing,*
> *1 $bCASH ≈ 1.0200 $CASH.*

This example is included for reference only and does not represent a guaranteed or fixed rate.

7.7 Implications for SAFV

Solido Grow demonstrates that:

- SAFV-style vaults can operate continuously in production
- standardized construction materially reduces complexity
- strategy-driven vaults can improve both yield and protocol stability

SAFV codifies these proven patterns into a reusable framework for the broader Supra ecosystem.

8. Roadmap and Future Extensions

SAFV is designed as a foundational framework that can evolve alongside the Supra AutoFi ecosystem. This section outlines future directions without introducing new assumptions or altering the framework's core guarantees.

8.1 No-Code Strategy Construction

A primary extension of SAFV is the development of no-code strategy construction tools.

Rather than modifying SAFV's execution or safety model, no-code tooling will operate as a configuration layer that:

- composes existing strategy templates,
- parameterizes predefined logic,
- deploys SAFV-compliant vaults without requiring direct contract development.

This preserves auditability and determinism while lowering the barrier to entry for strategists.

8.2 Strategy Templates and Standard Libraries

SAFV enables the creation of audited strategy templates for common use cases, such as:

- index allocation,
- delta-neutral yield,
- rule-based trading.

Standard libraries improve reuse, reduce implementation risk, and accelerate deployment without introducing discretionary behavior.

8.3 AI and External Signal Integration

SAFV is compatible with AI-generated or externally signed signals.
In all cases:

- signals remain advisory,
- strategies remain deterministic,
- vault constraints remain enforced.

This allows more sophisticated strategy design without compromising safety or execution guarantees.

8.4 Ecosystem Standardization

As adoption grows, SAFV may evolve into a shared construction standard through collaboration with ecosystem stakeholders.

Potential areas include:
- interface convergence,
- tooling interoperability,
- and shared audit assumptions.

Such evolution would occur incrementally and without altering existing deployments.

9. Conclusion

SAFV formalizes a construction framework for strategy-driven vaults built on top of Supra's AutoFi execution primitives. Rather than introducing new execution guarantees or financial instruments, the framework standardizes how existing primitives are composed into deterministic, auditable, and reusable on-chain products.

By separating custody, strategy logic, and execution, SAFV enables strategy-backed assets to be deployed without discretionary control or opaque behavior. The framework is derived from production systems and reflects practical constraints observed under real market conditions.

As automated finance continues to mature on Supra, SAFV provides a stable foundation for builders to experiment, iterate, and scale without fragmenting the ecosystem. Its design emphasizes early standardization, safety by construction, and long-term composability—principles that are essential for sustainable AutoFi adoption.

References

1. **Supra —** Supra AutoFi and AutoVault Documentation.
   Native automation, oracle-first execution, and MoveVM primitives forming the execution foundation for AutoVaults.
   https://supra.com

2. **Supra —** Supra Indices and On-Chain Data Feeds.
   Oracle-computed index feeds and real-time data aggregation used by strategy systems.
   https://supra.com/data/indices

3. **Ethereum Foundation —** ERC-4626: Tokenized Vault Standard.
   Foundational vault interface design informing SAFV's vault abstraction and accounting model.
   https://eips.ethereum.org/EIPS/eip-4626

4. **Move —** The Move Programming Language.
   Resource-oriented execution model underpinning safe asset custody and deterministic behavior on Supra.
   https://move-language.github.io/move/

5. **BlackRock —** Aladdin Risk Management Platform.
   Reference point for mandate-based, rule-driven portfolio execution in traditional finance.
   https://www.blackrock.com/aladdin

6. **BIS —** Principles for Financial Market Infrastructures (PFMI).
   Structural risk management concepts informing bounded execution and circuit-breaker design.
   https://www.bis.org

7. **Solido Money —** Solido Grow: Strategy-Driven Yield System on Supra.
   Production reference implementation from which SAFV abstractions are derived.
   (Open-source repositories and technical documentation to be published separately.) https://solido.money