BuilderAgent to Vault Integration Spec (v0.1)

BuilderAgent to Vault Integration Spec (v0.1)

Smart Contract Modules

- CuttlefishVault.sol - Holds DAO treasury funds (USDC, ETH, etc.)
- BuilderAgent.sol - AI-autonomous executor with strategy and trigger logic
- VaultPermissionRegistry.sol - Role and capability manager
- LiquidityAdapter.sol - Abstract interface for protocols (Uniswap, Lido, Balancer, etc.)

Core API Functions

1. requestVaultAllocation(uint256 amount, string memory strategyId)
- Purpose: BuilderAgent requests capital to execute a strategy
- Validation: Requires VAULT_EXECUTOR_ROLE
- Response: Emits VaultAllocationApproved or VaultAllocationRejected
- Notes: Triggers DAO vote if amount > THRESHOLD

2. submitTrade(bytes calldata tradeData, string memory strategyId)
- Purpose: Execute a swap or LP operation via LiquidityAdapter
- Validation: Must match registered strategy
- Side effects: Funds routed via CWALayer
- Logs: Trade timestamp, value, LP position ID

3. rebalanceLiquidity(string memory lpId)
- Purpose: Withdraw or redistribute LP positions based on new blueprint

- Guardrails: Throttle frequency or require vault signal consensus

- Notes: Emits VaultLPRebalanced(lpId, delta)

4. rollBackAction(bytes32 actionId)

- Purpose: Rollback a prior trade if ethics or safety checks fail

- Constraints: Only callable by AUDITOR_ROLE or with Chainlink trigger

Role Permissions Matrix

Role	Permissions
DAO_VOTER	Approve high-value trade proposals
VAULT_EXECUTOR	Request allocation, execute trades under threshold
ETHICS_VALIDATOR	Flag trades in conflict with constitution
AUDITOR_ROLE	Rollback failed or malicious agent actions
LIQUIDITY_OPERATOR	Register new LP strategies, adjust parameters
OWNER	Set thresholds, assign roles, upgrade logic

On-chain Logging Fields

Every vault interaction must include:

- actionType: "trade", "rebalance", "allocation"

- strategyId: ID of the triggering blueprint

- initiatedBy: Address (agent or human)

- ethicsScore: Optional; from validator agent

- vaultImpact: { delta: amount, token: USDC/ETH/etc. }