perplexity

Backbone Contract (`0x746dD4D401ce5Bbb0Fc964E1a7b4470619dBf67f`)

1. Purpose in the MountainShares Stack

The Backbone contract is the **central coordinator** that links all first-phase MountainShares components:

Capability	What Backbone Does	Connected Contract
Token minting	Calls the MountainShares token to mint PMS/EMS	`mountainSharesToken`
Customer purchases	Relays on-chain payments, tracks revenue	`customerPurchaseContract`
Heritage payouts	Splits NFT sale proceeds & logs revenue	`heritageContract`
Treasury oversight	Sends 20% of heritage income to monitor	`treasuryMonitor`
Business look-ups	Validates retailers & employers	`businessRegistry`
Price data	Reads oracle for USD-to-MS rates	`priceOracle`

2. Role-Based Security Model

Backbone inherits OpenZeppelin's `AccessControl` and defines four custom roles:

Role hash (bytes32)	Who Should Hold It	Powers
`DEFAULT_ADMIN_ROLE`	Governance multisig	Grants/revokes any role; can pause/unpause
`COORDINATOR_ROLE`	Automations / serverless bots	Executes the three "coordinate*" functions
`EMERGENCY_ROLE`	Security guardian	Triggers `emergencyPause` / `emergencyUnpause`
`HERITAGE_MANAGER_ROLE`	Curators or Clio backend	Routes heritage NFT revenue

All state-changing functions are **guarded** by one of these roles. Read-only getters remain permission-less.

3. Lifecycle Functions

1. `initializeSystem` (one-time, non-payable)
   Sets the six critical contract addresses, switches `systemInitialized` to `true`, and emits `SystemInitialized`.
2. `emergencyPause / emergencyUnpause`
   Toggles `systemPaused`. When paused, payable coordinator calls immediately revert, protecting funds during incidents.

3. `getSystemStatus`
   Returns a compact dashboard:

Field	Meaning
`initialized`	Has `initializeSystem` run?
`paused`	Is the contract halted?
`operations`	Total successful coordinator calls
`tokensMinted`	Aggregate MS minted through Backbone
`revenue`	Wei gathered and forwarded via the contract

4. Core Coordinator Methods

Function	State Mutability	High-Level Logic
`coordinateCustomerPurchase()`	`payable`	- Validates system not paused - Forwards ETH/USDC to purchase contract - Emits `OperationCoordinated("purchase", msg.sender, amount)`
`coordinateHeritageRevenue(address creator)`	`payable`	- Splits incoming payment 50 / 30 / 20 (creator / platform / treasury) - Emits both `RevenueDistributed` and `OperationCoordinated`
`coordinateTokenMint(address recipient,uint amount,string reason)`	`nonpayable`	- Mints MS via `mountainSharesToken` - Updates `totalTokensMinted` counter - Emits `TokensMinted` & `OperationCoordinated`

Each call increments `totalOperations`, creating a reliable audit trail.

5. Read-Only Dashboards

`getContractAddresses()` exposes all linked contracts in one call, simplifying front-end configuration.
Individual getters (`totalOperations`, `totalRevenue`, etc.) allow lightweight indexing.

6. Defensive Design Choices

- **Constructor Check:** Deploys with `initialized = false` and `systemPaused = true`, preventing accidents before setup.
- **Fallback & Receive:** Any unexpected direct send reverts unless explicitly handled by `receive()` (which simply logs revenue).
- **Interface Support:** Implements ERC-165; contracts and scripts can confirm Backbone's ABI with `supportsInterface`.

7. Typical Workflows

1. **First-Phase Purchase**
    1. Front-end pays ETH → `coordinateCustomerPurchase`.
    2. Purchase contract swaps to USDC & mints PMS.
    3. Backbone logs operation, updates revenue metrics.
2. **Heritage NFT Sale**
    1. Heritage contract forwards sale price to `coordinateHeritageRevenue`.
    2. 50% → creator, 30% → platform wallet, 20% → treasury.
    3. Events make accounting transparent for auditors and The Clio UI.
3. **Treasury-Initiated Mint**
    1. Treasury Monitor (with `COORDINATOR_ROLE`) calls `coordinateTokenMint`.
    2. Tokens minted to reserve, reason string logged for clarity.

8. Extension & Future Phases

- **Phase 2-3 hooks** can be added by granting new roles or upgrading linked contracts—Backbone's modular address registry supports plug-and-play upgrades.
- **AI-Driven Orchestration** (future): replace human `COORDINATOR_ROLE` holders with autonomous agents that respect pause checks and role limits.

9. Quick Strengths & Gaps

Strengths

- Centralized audit trail (`OperationCoordinated`)
- Fine-grained role control with admin hierarchy
- Built-in emergency shut-off
- Easy querying of system health and linked contracts

Gaps / Next Steps

- No native gas-refund or fee-sharing logic for coordinators
- Relies on external price oracle but doesn't validate staleness
- Upgradeability proxies not implemented; future upgrades require migration

Backbone's design gives MountainShares a **single point of orchestration** without becoming a single point of failure, thanks to robust role separation and pause mechanisms. It is the keystone that ensures every payment, mint, and heritage payout follows uniform, verifiable rules as the ecosystem scales beyond Phase 1.