Avalanche Pseudo-Delta-Neutral Leveraged Farming

This guide is intended for developers who wish to interact with Aperture contracts directly on Avalanche. Regular users are advised to use the official Aperture web app.

Please consult theDeployed Contracts page for contract addresses on various networks.

Query Position IDs For Wallet Address

Aperture uses position id to keep track of users' investment position. To get a list of position ids for an address, we need to first query Aperture Manager (seeDeployed Contracts for contract address).

JavaScript Python ```

Copy constmanagerABI= ["function getPositionsExt(address) view returns (tuple(uint128,uint16,uint64)[])"]; constapertureManager=newethers.Contract(addr,managerABI,provider); // getPositionsExt returns a list of tuple of: // uint128 positionId -> This is the position id. // uint16 chainId -> Used for cross chain communication. Likely not relevant for partner integration. // uint64 strategyId -> The strategy id (a.k.a. vault id) to identify which // strategy a position belongs to. constpositions=awaitapertureManager.getPositionsExt(ownerAddr);

Copy fromweb3importWeb3

w3=Web3(Web3.HTTPProvider('https://api.avax.network/ext/bc/C/rpc'))

manager_abi = '[{ "inputs": [{ "internalType": "address", "name": "user", "type": "address" }], "name": "getPositionsExt", "outputs": [{ "components": [{ "internalType": "uint128", "name": "positionId", "type": "uint128" }, { "internalType": "uint16", "name": "chainId", "type": "uint16" }, { "internalType": "uint64", "name": "strategyId", "type": "uint64" }], "internalType": "struct PositionInfoExt[]", "name": "", "type": "tuple[]" }], "stateMutability": "view", "type": "function" }]'

Look up manager address from Deployed Contract section of this doc.

contract=w3.eth.contract('0xeD380115259FcC9088c187Be1279678e23a6E565', abi=manager_abi) owner='0x6c3723d56bF091675FC7c327ee263a4D836E8055'

getPositionsExt returns a list of tuple of:

uint128 positionId -> This is the position id.

uint16 chainId -> Used for cross chain communication. Likely not relevant for partner integration.

uint64 strategyId -> The strategy id (a.k.a. vault id) to identify which

strategy a position belongs to.

print(contract.functions.getPositionsExt(owner).call())

``` Once we have position ids and strategy ids from Aperture Manager contract, we are ready to query individual vault to get detailed information for each position.

First step is to look up which vault to query against by using strategy id (seeDeployed Contracts for mapping). Next, look up position value expressed in AVAX value and convert it to be denominated by the input token of the vault.

JavaScript Python ```

Copy constvaultABI=[ "function vaultState() view returns (uint256, uint256)", "function getEquityETHValue() view returns (uint256)", "function pendingRewardStable() view returns (uint256)", "function positions(uint16,uint128) view returns
```

(uint256)" "function pairInfo() view returns (address, address, address, address)"];

constAVAXChainId=6;// Aperture uses Wormhole chain id. //getETHPx returns the price of a token in the native token. For Ethereum, it would be // ether. For Avalanche, it would be AVAX token. The final price is scaled to 2^112. constoracleABI= ["function getETHPx(address) view returns (uint256)"]; constvault=newethers.Contract(addr,vaultABI,provider); constoracle=newethers.Contract(addr,oracleABI,provider); // 2^112, a very large number used by oracle. constmultiplier=BigNumber.from(2).pow(112);

// Get total shares from the vault. constvaultShare=(awaitvault.vaultState())[0];// The first item is total vault shares. // Get how many shares a position owns. constpositionShare=awaitvault.position(AVAXChainId,positionId); // This is the token address user deposits into the vault. constmainTokenAddr=(awaitvault.pairInfo())[0];// First item is the main token address.

// Retrieve vault equity expressed in AVAX token. Note that this is different from TVL of // the vault. TVL includes leverage, however, equity is (total_position_worth - liability). // For example, equity of 1 is equivalent to 1 * leverage in terms of TVL of the vault. constequityETH=awaitvault.getEquityETHValue();

// Convert equity to be dominated in the main vault token.
constmainTokenETHPrice=awaitoracle.getETHPx(mainTokenAddr);
constpendingRewards=awaitvault.pendingRewardStable();

// Vault's equity expressed in main token.
constequityMainTotal=equityETH.mul(multiplier).div(mainTokenETHPrice).add(pendingRewards);
constequityMain=equityMainTotal.mul(positionShare).div(vaultShare); // Compute the final position value expressed in the main vault token. constresult=ethers.utils.formatUnits(equityMain,mainTokenDecimal);

Copy fromweb3importWeb3

w3=Web3(Web3.HTTPProvider('https://api.avax.network/ext/bc/C/rpc')) vault_abi = '[{ "inputs": [], "name": "vaultState", "outputs": [{ "internalType": "uint256", "name": "totalShareAmount", "type": "uint256" }, { "internalType": "uint256", "name": "lastCollectionTimestamp", "type": "uint256" }], "stateMutability": "view", "type": "function" }, { "inputs": [], "name": "getEquityETHValue", "outputs": [{ "internalType": "uint256", "name": "", "type": "uint256" }], "stateMutability": "view", "type": "function" },{ "inputs": [], "name": "pendingRewardStable", "outputs": [{ "internalType": "uint256", "name": "", "type": "uint256" }], "stateMutability": "view", "type": "function" },{ "inputs": [{ "internalType": "uint16", "name": "", "type": "uint16" }, { "internalType": "uint128", "name": "", "type": "uint128" }], "name": "positions", "outputs": [{ "internalType": "uint256", "name": "shareAmount", "type": "uint256" }], "stateMutability": "view", "type": "function" },{ "inputs": [], "name": "pairInfo", "outputs": [{ "internalType": "address", "name": "stableToken", "type": "address" }, { "internalType": "address", "name": "assetToken", "type": "address" }, { "internalType": "address", "name": "lpToken", "type": "address" }, { "internalType": "address", "name": "rewardToken", "type": "address" }], "stateMutability": "view", "type": "function" }]' oracle_abi = '[{ "inputs": [{ "internalType": "address", "name": "token", "type": "address" }], "name": "getETHPx", "outputs": [{ "internalType": "uint256", "name": "", "type": "uint256" }], "stateMutability": "view", "type": "function" }]' avax_chain_id=6

A large number used by oracle contract to scale up numbers.

multiplier=1<<112

Plug in desired position id.

position_id=173

See Deployed Contracts for detailed addresses break down.

vault=w3.eth.contract('0xa9a35E68dCC4aCA341Ccc21377A0340B1Ea1d163', abi=vault_abi)
oracle=w3.eth.contract('0xa6BAE2f3EE27271B55779BC6071FA101431Dc8Da', abi=oracle_abi)

vault_share=vault.functions.vaultState().call()[0] position_share=vault.functions.positions(avax_chain_id, position_id).call()
main_token_addr=vault.functions.pairInfo().call()[0]

equity denominated in native token.

equity_eth=vault.functions.getEquityETHValue().call()
main_token_eth_price=oracle.functions.getETHPx(main_token_addr).call()

pending_rewards=vault.functions.pendingRewardStable().call()

equity_main_total=equity_eth*multiplier//main_token_eth_price+pending_rewards*
*equity_position=equity_main_total*position_share//vault_share

```
```