Rari Capital Ethereum API Documentation

Welcome to the API docs for `RariFundManager`, `RariFundToken`, and `RariFundProxy`, the smart contracts behind Rari Capital's stablecoin fund. You can find out more about Rari at www.rari.capital.

uint256 RariFundToken.balanceOf(address account)

Returns the amount of RFT owned by `account`.

Parameters:

- `account` (address) - The account whose balance we are retrieving.

uint256 RariFundManager.balanceOf(address account)

Returns an account's total balance in USD (scaled by 1e18).

Parameters:

- `account` (address) - The account whose balance we are calculating.

int256 RariFundManager.interestAccruedBy(address account)

Returns the total amount of interest accrued by `account` (excluding the fees paid on interest) in USD (scaled by 1e18).

Parameters:

- `account` (address) - The account whose interest we are calculating.

Development notes:

- *Ideally, we can add the view modifier, but Compound's `getUnderlyingBalance` function (called by `getRawFundBalance`) potentially modifies the state.*

Development notes:

- *Ideally, we can add the view modifier, but Compound's `getUnderlyingBalance` function (called by `getRawFundBalance`) potentially modifies the state.*

bool RariFundManager.isCurrencyAccepted(string currencyCode)

Returns a boolean indicating if deposits in `currencyCode` are currently accepted.

Parameters:

- `currencyCode` (string): The currency code to check.

bool RariFundProxy.deposit(string currencyCode, uint256 amount)

For a limited time only, we are paying gas fees for first-time deposits of at least 250 DAI!

Deposits funds to RariFund in exchange for RFT (with GSN support).

You may only deposit currencies accepted by the fund (see `RariFundManager.isCurrencyAccepted(string currencyCode)`).

Please note that you must approve RariFundProxy to transfer at least `amount`.

Parameters:

- `currencyCode` (string): The currency code of the token to be deposited.
- `amount` (uint256): The amount of tokens to be deposited.

Return value: Boolean indicating success.

bool RariFundManager.deposit(string currencyCode, uint256 amount)

Deposits funds to RariFund in exchange for RFT.

You may only deposit currencies accepted by the fund (see `RariFundManager.isCurrencyAccepted(string currencyCode)`). However, `RariFundProxy.exchangeAndDeposit` exchanges your funds via 0x and deposits them in one transaction.

Please note that you must approve RariFundManager to transfer at least `amount`.

Parameters:

- `currencyCode` (string): The currency code of the token to be deposited.
- `amount` (uint256): The amount of tokens to be deposited.

Return value: Boolean indicating success.

bool RariFundManager.withdraw(string currencyCode, uint256 amount)

Withdraws funds from RariFund in exchange for RFT.

You may only withdraw currencies held by the fund (see `RariFundManager.getRawFundBalance(string currencyCode)`). However, `RariFundProxy.withdrawAndExchange` withdraws your funds and exchanges them via 0x in one transaction.

Please note that you must approve RariFundManager to burn of the necessary amount of RFT.

Parameters:

- `currencyCode` (string): The currency code of the token to be withdrawn.
- `amount` (uint256): The amount of tokens to be withdrawn.

Return value: Boolean indicating success.

bool RariFundProxy.exchangeAndDeposit(address inputErc20Contract, uint256 inputAmount, string outputCurrencyCode, LibOrder.Order[] orders, bytes[] signatures, uint256 takerAssetFillAmount)

Exchanges and deposits funds to RariFund in exchange for RFT.

You can retrieve order data from the 0x swap API. See the web client for implementation.

Please note that you must approve RariFundProxy to transfer at least `inputAmount` unless you are inputting ETH. You also must input at least enough ETH to cover the protocol fee (and enough to cover `orders` if you are inputting ETH).

Parameters:

- `inputErc20Contract` (address): The ERC20 contract address of the token to be exchanged. Set to address(0) to input ETH.
- `inputAmount` (uint256): The amount of tokens to be exchanged (including taker fees).
- `outputCurrencyCode` (string): The currency code of the token to be deposited after exchange.
- `orders` (LibOrder.Order[]): The limit orders to be filled in ascending order of the price you pay.
- `signatures` (bytes[]): The signatures for the orders.
- `takerAssetFillAmount` (uint256): The amount of the taker asset to sell (excluding taker fees).

Return value: Boolean indicating success.

Development notes:

- *We should be able to make this function external and use calldata for all parameters, but Solidity does not support calldata structs.*

bool RariFundProxy.withdrawAndExchange(string[] inputCurrencyCodes, uint256[] inputAmounts, address outputErc20Contract, LibOrder.Order[][] orders, bytes[][] signatures, uint256[] makerAssetFillAmounts, uint256[] protocolFees)

Exchanges and deposits funds to RariFund in exchange for RFT.

You can retrieve order data from the 0x swap API. See the web client for implementation.

Please note that you must approve RariFundManager to burn of the necessary amount of RFT. You also must input at least enough ETH to cover the protocol fees.

Parameters:

- `inputCurrencyCodes` (string[]): The currency codes of the tokens to be withdrawn and exchanged.
- `inputAmounts` (uint256[]): The amounts of tokens to be withdrawn and exchanged (including taker fees).
- `outputErc20Contract` (address): The ERC20 contract address of the token to be outputted by the exchange. Set to address(0) to output ETH.
- `orders` (LibOrder.Order[][]): The limit orders to be filled in ascending order of the price you pay.
- `signatures` (bytes[][]): The signatures for the orders.
- `makerAssetFillAmounts` (uint256[]): The amounts of the maker assets to buy.
- `protocolFees` (uint256[]): The protocol fees to pay to 0x in ETH for each order.

Return value: Boolean indicating success.

Development notes:

- *We should be able to make this function external and use calldata for all parameters, but Solidity does not support calldata structs.*

uint256 RariFundManager.getFundBalance()

Returns the fund's total investor balance (all RFT holders' funds but not unclaimed fees) of all currencies in USD (scaled by 1e18).

Development notes:

- *Ideally, we can add the view modifier, but Compound's `getUnderlyingBalance` function (called by `getRawFundBalance`) potentially modifies the state.*

int256 RariFundManager.getInterestAccrued()

Returns the total amount of interest accrued by past and current RFT holders (excluding the fees paid on interest) in USD (scaled by 1e18).

Development notes:

- *Ideally, we can add the view modifier, but Compound's `getUnderlyingBalance` function (called by `getRawFundBalance`) potentially modifies the state.*

uint256 RariFundManager.getInterestFeeRate()

Returns the fee rate on interest.

int256 RariFundManager.getInterestFeesGenerated()

Returns the amount of interest fees accrued by beneficiaries in USD (scaled by 1e18).

Development notes:

- *Ideally, we can add the view modifier, but Compound's `getUnderlyingBalance` function (called by `getRawFundBalance`) potentially modifies the state.*

int256 RariFundManager.getRawInterestAccrued()

Returns the raw total amount of interest accrued by the fund as a whole (including the fees paid on interest) in USD (scaled by 1e18).

Development notes:

- *Ideally, we can add the view modifier, but Compound's `getUnderlyingBalance` function (called by `getRawFundBalance`) potentially modifies the state.*

uint256 RariFundManager.getRawFundBalance()

Returns the fund's raw total balance (all RFT holders' funds + all unclaimed fees) of all currencies in USD (scaled by 1e18).

Development notes:

- *Ideally, we can add the view modifier, but Compound's `getUnderlyingBalance` function (called by `getRawFundBalance`) potentially modifies the state.*

uint256 RariFundManager.getRawFundBalance(string currencyCode)

Returns the fund's raw total balance (all RFT holders' funds + all unclaimed fees) of the specified currency.

Parameters:

- `currencyCode` (string): The currency code of the balance to be calculated.

Development notes:

- *Ideally, we can add the view modifier, but Compound's `getUnderlyingBalance` function (called by `RariFundController.getPoolBalance`) potentially modifies the state.*

uint256 RariFundController.getPoolBalances(string currencyCode)

Returns the fund controller's balance of each pool of the specified currency.

Parameters:

- `currencyCode` (string): The currency code whose balance is to be calculated.

Return value: An array of pool indexes and an array of corresponding balances.

Development notes:

- *Ideally, we can add the view modifier, but Compound's `getUnderlyingBalance` function (called by `getPoolBalance`) potentially modifies the state.*