Formalizing ERC-4626

Transmissions11

February 2022

Abstract

We formalize the behavior of the ERC-4626 specification, prove certain assumed properties hold over all possible actions, and provide recommendations on implementing the specification in a language without floating point arithmetic.

Contents

1 Introduction

ERC-4626 is a standardized interface and set of behaviors for *Vault-like* smart contracts on Ethereum to implement and integrate into their systems.

A Vault is a smart contract that accepts deposits of a specific *asset* (typically an *ERC-20* token) from users. Users who deposit into a Vault receive an equivalent amount of *shares*, which represent a claim on the assets they deposited. Shares are typically "tokenized" in the form of an ERC-20 token. The Vault is free to manipulate the assets deposited into it, lending them out for yield, locking them up for time, etc.

As the Vault accrues interest, the value of its depositor's shares increase in tandem. Depositors can then redeem their shares and receive their principal along with a proportional share of the Vault's profits. This pattern is quite common, but implementations vary slightly from case-to-case. ERC-4626 was proposed with the goal of standardizing this basic pattern, making it easier for other applications to interact with Vault-like systems.

A high-level summary of the ERC can be found in many places, but in this paper we seek formalize the low-level properties of a basic ERC-4626 implementation, to serve as a reference for developers wishing to implement the standard optimally.

2 Definitions

Let σ be the total assets held by an ERC-4626 Vault. A specific quantity of assets will be represented by σ_n. When discussing specific actions users can take with a vault like *deposit*, we use σ to represent the total assets before the action, and σ' to represent the total assets held after the action, like so:

$$\sigma' = \sigma + \sigma_n$$

Similarly, let μ be the total supply of shares, while a specific quantity will be represented by μ_n. An updated quantity of shares is represented by μ', like so:

$$\mu' = \mu + \mu_n$$

A specific user's balance of shares will be represented as μ_i, and an updated balance will be written as μ_i', like so:

$$\mu_i' = \mu_i + \mu_n$$

3 Deposit

The *deposit* function takes an amount of underlying tokens (σ_n), and mints the user a specific quantity of shares (μ_n) representing a claim on their deposit.

3.1 State Transitions

We will begin by formalizing the state transitions that occur during a *deposit*.

3.1.1 User Shares

The user is minted an amount of shares proportional to their deposit:

$$\mu_i' = \mu_i + \mu_n$$

Computing Shares To Mint To determine μ_n, we must first define a formula to compute the rate of exchange between assets and shares. Using only the total supply of shares (μ) and total assets held by the Vault (σ), we can deduce that the rate of shares to assets is:

$$\frac{\mu}{\sigma}$$

Using this ratio, we can determine the amount of shares (μ_n) a σ_n assets is worth with the following formula:

$$\mu_n = \sigma_n \times \frac{\mu}{\sigma}$$

However, in languages without floating point arithmetic, this formula will yield faulty results. To improve precision and avoid early truncation, we simplify the formula like so:

$$\mu_n = \frac{\sigma_n \mu}{\sigma}$$

In this simplified form, it is trivial to confirm the correctness of the expression using dimensional analysis. The quantities of assets cancel each other out, leaving us with a quantity of shares, our desired result.

3.1.2 Total Shares

Returning to our discussion of the state transitions that occur after a *deposit*, we note that the total supply of shares will increase by the amount of shares minted to the user (μ_u), which we just described how to compute.

$$\mu' = \mu + \mu_u$$

3.1.3 User Assets

The user's balance of assets decreases by the amount of assets they are depositing.

$$\sigma_i' = \sigma_i - \sigma_n$$

3.1.4 Total Assets

The total balance of assets increases by the amount of tokens deposited:

$$\sigma' = \sigma + \sigma_n$$

3.2 Properties

With the function's state transitions formalized, we can now prove some key properties hold after a *deposit*.

3.2.1 Exchange Rate Persistence

We begin our proof by defining the high level statement we are asserting: the ratio of assets to shares does not change after a *deposit*.

$$\frac{\sigma}{\mu} = \frac{\sigma'}{\mu'}$$

Expanding with the defined state transitions formalized above, we can rewrite this expression like so:

$$\frac{\sigma}{\mu} = \frac{\sigma + \sigma_n}{\mu + \mu_n}$$

Further expanding with our definition of μ_n:

$$\frac{\sigma}{\mu} = \frac{\sigma + \sigma_n}{\mu + \frac{\sigma_n \mu}{\sigma}}$$

Now we can condense:

$$\frac{\sigma}{\mu} = \frac{\sigma + \sigma_n}{\frac{\sigma\mu + \sigma_n\mu}{\sigma}} = \frac{\sigma^2 + \sigma\sigma_n}{\mu\sigma + \sigma_n\mu} = \frac{\sigma(\sigma + \sigma_n)}{\mu(\sigma + \sigma_n)} = \frac{\sigma}{\mu}$$

Therefore we can conclude that the rate of exchange between shares and assets ($\frac{\sigma}{\mu}$) remains constant after a deposit of any size, or put simply, the assertion holds true.

$$\frac{\sigma}{\mu} = \frac{\sigma'}{\mu'}$$

4 Redeem

The *redeem* function takes an amount of shares (μ_n), and burns them, in exchange for transferring the user an equivalent amount of assets (σ_n) their shares were redeemable for.

4.1 State Transitions

The state transitions that occur during a *redeem* can be thought of as the inverse of the transitions that occur during a *deposit*.

4.1.1 User Shares

The user has their balance of shares decreased by the amount of shares being redeemed:

$$\mu_i' = \mu_i - \mu_n$$

4.1.2 Total Shares

The total supply of shares will decrease by the amount of shares being redeemed:

$$\mu' = \mu - \mu_u$$

4.1.3 User Assets

The user's balance of assets increases by the amount of assets equivalent to the value of their shares:

$$\sigma_i' = \sigma_i + \sigma_n$$

Computing The Value Of Shares Computing the value of an amount of shares μ_n, is simple:

$$\sigma_n = \frac{\mu_n \sigma}{\mu}$$

Using basic dimensional analysis, we can confirm this formula is correct, by observing that the quantity of shares in the numerator are canceled out by the total shares in the denominator, leaving us with a quantity of assets.

It is again important to note that the expression has been purposely condensed to avoid early truncation in a language without floating point arithmetic. In these languages it is critical to ensure multiplication is done before division wherever possible.

4.1.4 Total Assets

The total balance of assets decreases by the amount of assets equivalent to the amount of shares burned (σ_n), which was computed above:

$$\sigma' = \sigma - \sigma_n$$

4.2 Properties

With the function's state transitions formalized, we can now prove some key properties hold after a *redeem*.

4.2.1 Exchange Rate Persistence

We begin our proof by defining the high level statement we are asserting: the ratio of assets to shares does not change after a *redeem*.

$$\frac{\sigma}{\mu} = \frac{\sigma'}{\mu'}$$

Expanding with the defined state transitions formalized above, we can rewrite this expression like so:

$$\frac{\sigma}{\mu} = \frac{\sigma - \sigma_n}{\mu - \mu_n}$$

Further expanding with our definition of σ_n:

$$\frac{\sigma}{\mu} = \frac{\sigma - \frac{\mu_n \sigma}{\mu}}{\mu - \mu_n}$$

Now we can condense:

$$\frac{\sigma}{\mu} = \frac{\frac{\sigma\mu - \mu_n\sigma}{\mu}}{\mu - \mu_n} = \frac{\sigma\mu - \mu_n\sigma}{\mu^2 - \mu_n\mu} = \frac{\sigma(\mu - \mu_n)}{\mu(\mu - \mu_n)} = \frac{\sigma}{\mu}$$

Therefore we can conclude that the rate of exchange between shares and assets ($\frac{\sigma}{\mu}$) remains constant after a redemption of any size, or put simply, the assertion holds true.

$$\frac{\sigma}{\mu} = \frac{\sigma'}{\mu'}$$