MELON PROTOCOL: A PROTOCOL FOR DIGITAL ASSET MANAGEMENT STRATEGIES DRAFT

RETO TRINKLER AND MONA EL ISA

Abstract. The Melon protocol is a protocol for digital asset management on the Ethereum platform. It enables participants to set up, manage and invest in digital asset management strategies in an open, competitive and decentralised manner.

1. Introduction

The value and importance of a wide range of digital assets[1] has risen dramatically over the last few years. Hence the question naturally arises how to manage this new and fast-growing asset class in the most advantageous way.

This could be done by investing in a hedge fund which specializes in digital assets.

However, the lack of standardisation can make comparison of fund performances difficult and fund auditing practices can be very opaque.

A second deterrent, is the time and high cost required for setting up and running a hedge fund. This limits the range of possible hedge fund managers to a comparatively small group of people. It arguably also limits competition between hedge funds creating qualitatively less good hedge fund performances[14].

A third deterrent to investing in a hedge fund is some of the outdated technological infrastructure, giving room for a lot of inefficiencies.

Section 2 through 4 of this paper will discuss the general mechanics of the Melon protocol. Section 5 through 7 will discuss how the Melon protocol can be used to solve above objectives of openness, competitiveness and security. Finally, section 8 will propose a solution for protocol development.

2. Assets

To better understand the general mechanics of the Melon protocol, lets start by defining the term *digital asset management strategy*. A digital asset management strategy is, in the context of this paper, regarded as a strategy on how to manage a *portfolio*[2]. Each portfolio can hold a variety of digital assets, where these digital assets represent the value a portfolio can hold.

As a motivation to the challenge ahead, let's look at a few examples of these digital assets currently available, or in development.

For the sake of ease, and to highlight the differentiation in underlying value, we categorise digital assets into three sets: Digital assets which gain their value from collateralisation of an underlying asset, called *collateralised assets*. Digital assets which do not gain their value from collateralisation, called *un-collateralised assets*. Finally, digital assets which are derived from existing digital assets called *derivatives*.

2.1. **Collateralised Assets.** Collateralised Assets, are assets which gain their value from the collateralisation of *real-world assets*. Examples are Dai from the Dai Credit System[13], Dassets[11] from String Technology or t0[12] from Overstock. An example for a digital asset backed by a commodity is DGX from Digix[4] which binds the value of gold to a digital asset.

2.2. **Un-collateralised Assets.** Un-collateralised assets are digital assets which gain their value in the scarcity of the token itself. Examples for un-collateralised assets are ETH (Ethereum[7]), ETC (Ethereum Classic[6]), REP (Augur[1]), DGD (Digix[4]) or MKR (Maker[13]). Eventually even companies issuing tokens as shares on the Blockchain will belong to this set.

2.3. **Derivatives.** The third set of digital assets are derivatives of other digital assets. In the context of this work, a *derivative* is defined as a digital asset which has its value directly derived from another digital asset. An example for this set is a contract for difference (CFD) of an existing digital asset.

In conclusion, the restriction to digital assets will be with the adoption of Ethereum and the various decentralised applications and services built upon it be less and less restrictive.

3. Portfolio

Having seen the selection of digital assets waiting to be managed by up-and-coming Portfolio Managers, let's now discuss the general mechanics of how this can be achieved technically.

Each portfolio consists of a *core* part and a set of *modules* (see figure 1)

3.1. **Core.** The core part, written in a set of smart-contracts, can be seen as the part which holds everything together. The modules, also written in a set of smart-contracts, can be seen as the auxiliary functionality to the core part. The core part together with a set of rules

E-mail addresses: (Reto Trinkler) rt@melonport.com, (Mona El Isa) me@melonport.com.

[1]Throughout the present work, a *digital asset*, is regarded as a digital token of value, run and stored on the Ethereum Blockchain. There is no technical difference assumed between what an *asset* and what a *token* is, the terms are interchangeable. However in the context of portfolio management usually the term *asset* is used.

[2]Throughout the present work, a *portfolio* is regarded as a finite set of digital assets.

on how the core interacts with its modules constitute the *Melon protocol*.

3.2. **Modules.** The modules provide auxiliary functionality to the core part. For example, providing off-chain data, storing of portfolio relevant data or the executing of calculations for the portfolio.

The modules are the parts of the portfolio which have subjective functionality. Through the modular concept of the portfolio, these subjective parts can be changed. For example different Portfolio Managers might want to charge different management fees and apply their own methods on how to calculate these fees. The modular concept allows the managers to chose the management fee module of their liking. Thus they can select the fees and calculation methodology by simply linking the corresponding module to the core.

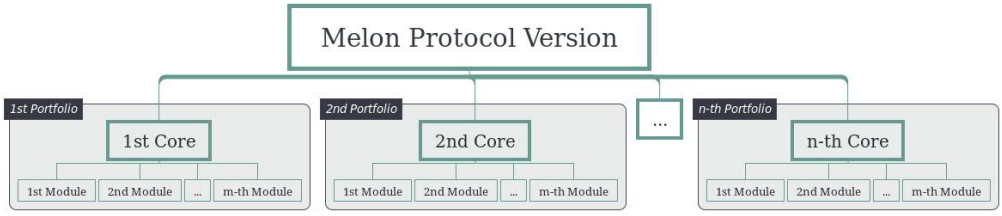

FIGURE 1. Protocol version links existing portfolios and collects licensing fees

Modules can be freely developed by anyone. Developers can then define a commission that is paid back to them every time someone uses their module.

The Melon protocol includes the following modules as a foundation:

Registrar: Links assets to price feeds to trading.
Functionality: A way to build assets specific functionality.
Price Feeds: Provide a price for an asset by taking one or several data sources into account.
Exchanges: On-chain marketplaces such as EtherEx[8] or Maker-Market[10], where one or a set of assets can be traded.
Trading: A set of rules on how trading is permitted.
Management Fee: A gross asset value (see appendix C) independent payment for the Portfolio Manager.
Performance Fee: A gross asset value (see appendix C) dependent payment for the Portfolio Manager.

In the *registrar module*, the Portfolio Manager selects a finite amount of assets as well as a finite amount of exchanges on which these assets can be traded. All a Portfolio Manager can ever do is trade those specific assets on those specific exchanges. The smart contracts do not allow funds to be sent in any other way or to any other accounts.

The *functionality module* allows a Portfolio Manager to retain *actions* or *rights* and avoid penalization from non-action (eg. Augur's REP tokens). These interactions between token custodian and corresponding smart-contracts need to be programmed in as it is no longer an individual's or non-contract account which controls these funds.

The *price feeds module* is needed as asset prices, in general, tend to be different on different exchanges. This is because of the inherent way prices are set - a price is set by demand and supply which constitutes of market participants wanting to buy or sell. In this context, the word *market participants* also refers to trading algorithms, etc. In general, market participants reflect a different demand and supply profile on different exchanges generating different bid/ask prices. While it's true that arbitrageurs keep the differences small, they remain, for at least as long as there is a fee to be paid on the execution of trades. A fee might be a commission to the exchange or broker, or a fee in form of another execution cost, such as for example the gas cost to be paid on the Ethereum network. To solve the problem of ambiguous prices, the Melon protocol will require the Portfolio Manager to choose one price feed module providing one specific price against which the assets under management (AUM) are evaluated.

The *exchange module* specifies an on-chain exchange where assets can be traded. This is required as it poses a restriction on where the Portfolio Manager can trade his/her funds. See also registrar module.

The *trading module* restricts trading and links to a preselected exchange on which an asset can be traded. For example, no trade size can be higher than 10% of the volume traded of this asset. This module is intended to reduce the amount of order book manipulation in favour of the Portfolio Manager.

A Portfolio Manager will be able to receive a management fee as well as a performance fee. These fees are specified in the *management fee module* and the *performance fee module*. The management fee is generally calculated by reference to the assets under management, i.e. the gross asset value (see appendix C) of the portfolio. The performance fee is generally calculated by reference to the *increase* in the gross asset value. There is usually a high water mark, which means that if the portfolio performing below a defined benchmark since inception, the Portfolio Manager does not get paid a performance fee. Alternative forms of performance fees are conceivable. For example, the interval by which the performance fee gets paid out can be selected by the Portfolio Manager.

Before the deployment of a portfolio, a Portfolio Manager decides which modules he/she would like to use.

Once the portfolio is deployed this then can be seen as the offering of a legal contract. The contract terms are unambiguously visible and securely held by the Blockchain.

Therefore, the *smart-contract terms* are agreed upon by the investor upon investing in the portfolio.

4. Investing and Redeeming

There are two ways to invest in a portfolio. The first way is to buy *shares* of the portfolio on any marketplace on which they are traded. The second way is to *create* shares by investing Ether directly into the respective smart contract of the portfolio.

Similar to investing there are three ways to redeem from a portfolio. The first way is to sell *shares* of the portfolio on any marketplace on which they are traded. And the second way is to *annihilate* shares. This can be done by

- redeeming into a separate portfolio or
- redeeming directly into Ether via a program trade[3].

4.1. Net Asset Value.

Shares of a portfolio are designed such that they fulfil the following properties:

- Shares are fungible.
- Shares reflect ownership of the portfolio.
- The inherent value of the shares is given by the value of the underlying assets of the portfolio.
- Share price is *relatively* independent of investments and withdrawals made.

Shares will be represented by a smart-contract following the Ethereum token standard[5]. Thus they are fungible and tradable on exchanges such as EtherEx[8] or Maker-Market[10]. Furthermore, Portfolio Managers can hold and manage shares of other portfolios.

Shares also reflect ownership of the portfolio, where the formula of ownership is the following:

$$Ownership = \frac{Shares\ holding}{Total\ shares\ in\ existence} \tag{1}$$

For example if one holds ten out of a hundred total shares then the ownership is 0.1 or 10%. This is dynamic, as people invest and redeem the total amount of shares in existence changes and therefore also the percentage of ownership.

The inherent share price (see appendix G) is *defined* by the net asset value per share (see appendix F) and thus by the value of the underlying assets.

Every time an Investor invests Ether in a portfolio, shares are created and every time an Investor redeems Ether, shares are exchanged against the value of the underlying assets and thereby annihilated. This mechanism of creating and annihilating shares keeps the share price *relatively* independent of investments respective to withdrawals made.

In conclusion, the inherent share price is set, not by demand, but by performance. Each investor has the ability to exchange its shares at any time against the value of the underlying assets.

4.2. Creation.

By investing funds F denominated in Ether, into a portfolio, shares are created. The Investor sends these funds directly to the smart contract of the portfolio where they are added to the portfolio $\underline{h}_m^{t_i}$ (see appendix B):

$$\begin{pmatrix} h_{a_1}^{t_i} \\ h_{a_2}^{t_i} \\ \vdots \\ h_{a_n}^{t_i} \end{pmatrix} \rightarrow \begin{pmatrix} h_{a_1}^{t_i} + F \\ h_{a_2}^{t_i} \\ \vdots \\ h_{a_n}^{t_i} \end{pmatrix}$$

Where by convention $h_{a_1}^{t_i}$ is the amount of Ether the portfolio m holds, at time t_i.

Then the *quantity of shares* $q_F^{t_i}$ created for funds F, at time t_i is calculated. The formula is the following:

$$q_F^{t_i} = \frac{F}{p_m^{t_i}} \tag{2}$$

Note, by definition the share price $p_m^{t_i}$ is independent of Funds F invested.

The quantity of shares $q_F^{t_i}$ are then allocated to the Investor. For the investment to get incorporated takes at least one block time, i.e. more than 10 to 19 seconds. During this time period the share price might change. However the Investor will be able to invest via a limit order hence they have no risk of getting an unexpected price.

4.3. Annihilation.

By redeeming funds F denominated in Ether, from a portfolio, shares are annihilated. The investor redeems funds F by exchanging shares against the underlying value they represent.

To request a withdrawal of amount F, one has to withdraw:

$$q_F^{t_i} = \frac{F}{p_m^{t_i}} \quad \left(\Leftrightarrow F = p_m^{t_i} q_F^{t_i} \right) \tag{3}$$

of shares.

This quantity of shares $q_F^{t_i}$ represents a percentage of ownership, called $o_F^{t_i}$. The formula for $o_F^{t_i}$ is the following:

$$o_F^{t_i} = \frac{q_F^{t_i}}{Total\ shares\ in\ existence} \tag{4}$$

The Investor now redeems this percentage of all the assets a portfolio holds, directly from the smart contract of the portfolio. In doing so he/she sets up a new portfolio. The new portfolio of the Portfolio Manager is thus:

$$\begin{pmatrix} h_{a_1}^{t_i} \\ h_{a_2}^{t_i} \\ \vdots \\ h_{a_n}^{t_i} \end{pmatrix} \rightarrow (1 - o_F^{t_i}) \begin{pmatrix} h_{a_1}^{t_i} \\ h_{a_2}^{t_i} \\ \vdots \\ h_{a_n}^{t_i} \end{pmatrix}$$

Where the separated part, is the part which belongs to the Investor, redeeming funds:

$$o_F^{t_i} \begin{pmatrix} h_{a_1}^{t_i} \\ h_{a_2}^{t_i} \\ \vdots \\ h_{a_n}^{t_i} \end{pmatrix}$$

This separated part can now be seen as the portfolio of the investor. The investor becomes its own Portfolio Manager. They can now either decide to manage this new portfolio to their liking or liquidate the assets to Ether through a program trade. A program trade is an algorithm that liquidates a portfolio to complete the redeeming process.

Since portfolios separate when redeeming funds, one cannot directly expect or anticipate any future trades

[3] A program trade is a trade where orders are entered directly into the market and executed automatically.

made by the redeeming investor. All one can see on the Blockchain is the separating of the portfolio. This process can be done in a simple and convenient way, where the investor chooses his/her selling off strategy on an off-chain server which listens to the Blockchain and trades accordingly.

4.4. **Open and closed-ended portfolios.** Generally one differentiates between two types of portfolios. *open-ended portfolios* and *closed-ended portfolios*. The difference between the two is that in the former there's no limit on how much investors can invest while in the latter there is. In a closed-ended portfolio, once the investment limit of the portfolio has been reached the process of creation will be suspended. At this point, shares can only be bought on exchanges.

In conclusion, the Investor is always in control of their investment and can exchange back shares to get the underlying value of the assets without having to ask for permission from the Portfolio Manager or anybody else.

5. Open

As seen in section 4, portfolio share prices are defined by the net asset value per share. This is true for each portfolio deployed, meaning that share price of all Melon protocol portfolios are visible and comparable on the Blockchain. The Portfolio Manger's managing and trading track-record is visible and auditable in the same way.

In addition, all of the Melon protocols' smart contracts are open-source[4].

In conclusion, the Melon protocol is an open-source protocol. Portfolio track-records and performances are visible and auditable by everyone.

6. Competitive

The competitive gains of the Melon protocol are in the form of lower cost and time barriers to setting up and running a portfolio.

The costs and complexity to setting up a portfolio using the Melon protocol are lower than they are with traditional asset management, seconds and cents versus months and millions. Such benefits will favour all investors, but especially large scale money managers for whom the majority of operating costs will be eliminated (well over 50% of a typical asset management firm's costs today are made up of fund administration and operations infrastructure[9]). These radical cost savings can be passed on to savers. The lower operating costs will also enable new up-and-coming Portfolio Managers to enter the market by reducing minimum scale requirements and start up costs.

The cost of running a portfolio on the Blockchain is equal to the core licensing fees, modular commissions and the infrastructure costs to be paid on the Ethereum platform (see Figure 2).

The licensing fees are set by the protocol and the modular fees are set by the module developers. Both of them are expected to be a fraction of a cent for each usage.

The infrastructure costs are equal to the *gas* used for the execution of the underlying smart contracts. The gas costs are dependent on the *gas price* set for the transactions and the *amount of gas* used in executing these smart contracts[15].

In conclusion, by having low set up requirements and low costs of running a portfolio one can create a never-seen-before competitive environment for asset management strategies.

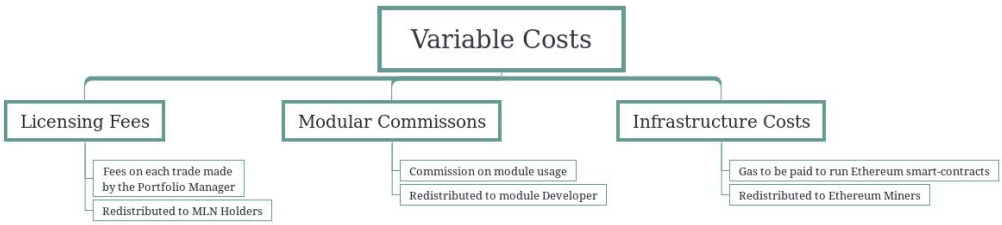

Figure 2. Variable Costs of a Melon protocol portfolio

7. Decentralised

A Melon protocol portfolio can be *set up*, *managed* and *invested in*, using a decentralised technological infrastructure relying on the Ethereum Blockchain.

In this context, one can differentiate between *decentralised storage* and *decentralised execution*.

7.1. **Decentralised Storage.** All of the Melon protocol's smart-contracts, portfolio track records and assets are stored on a decentralised Blockchain.

Storing smart-contracts and portfolio track records in a decentralised way mitigates the risks around single points of failure and provides open and reliable storage of information.

Storing portfolio assets in a decentralised way, reduces custody risks.

Incidents like the financial crisis of 2008 and the 2013 bank deposit levy in Cyprus have taken a heavy toll on the trust of centralised custodians. Legislation of *bail-ins* in many developed countries doesn't help either[2][3].

7.2. **Decentralised Execution.** Execution of the smart-contracts is done in a decentralised manner using the Ethereum virtual machine (EVM) which is distributed onto all nodes connected to the Ethereum network. The

[4]They are published under `https://github.com/melonproject`.

result is generally more efficiency, security and predictability. Most notably, counterparty and settlement risks of trades are reduced significantly.

In conclusion, by having decentralised storage and execution one can mitigate some of the potential security vulnerabilities and market inefficiencies, such as reducing single points of failure, custody-, counterparty-, and settlement risks.

8. Protocol Development

To build the Melon protocol and strengthen the network effect, there will be a digital token issued. This token is called the *Melon token* (MLN) and will be distributed through a crowd-sale.

Through the usage of the core, licensing fees are generated. These are collected and proportionally redistributed back to the Backers of the Melon protocol, i.e. MLN Holders.

9. Future Directions

To drive adoption, there will be portals[5]. These are user-friendly web applications to access the Melon protocol. This will allow users to interact with the protocol easily.

The value of the protocol is directly correlated with the value of these portals. The better and easier the portals, the more licensing fees will be collected and redistributed.

10. Conclusion

The Melon protocol proposed a protocol for digital asset management on the Ethereum platform. It enables participants to set up, manage and invest in digital asset management strategies in an open, competitive and decentralised manner.

11. Acknowledgements

We would like to use this opportunity to express our gratitude to everyone who supported us throughout the course of writing this paper.

A special thanks to Garrett Cassidy, André Wolke, Roman Bischoff, Jorge Mielgo, Sandro Lera, Dylan Grice and Andrey Ternovsky for their support, feedback and improvements to this paper.

References

[1] Augur - decentralized prediction market. http://www.augur.net/. Accessed: 2016-08-26.

[2] Beteiligung der gläubiger an der bankenrettung. https://www.finma.ch/de/ueberwachung/banken-und-effektenhaendler/aufsichtsinstrumente/stabilisierungs-und-abwicklungsplanung/kapitalmassnahmen/. Accessed: 2016-08-26.

[3] Deal reached on bank "bail-in directive". http://www.europarl.europa.eu/news/en/news-room/20131212IPR30702/Deal-reached-on-bank-%E2%80%9Cbail-in-directive%E2%80%9D. Accessed: 2016-08-26.

[4] Digix platform. https://www.dgx.io. Accessed: 2016-08-26.

[5] Ethereum - token standard. https://github.com/ethereum/EIPs/issues/20. Accessed: 2016-08-26.

[6] Ethereum classic. https://ethereumclassic.github.io/. Accessed: 2016-08-26.

[7] Ethereum platform. https://ethereum.org/. Accessed: 2016-08-26.

[8] Etherex - decentralized exchange. https://etherex.org/. Accessed: 2016-08-26.

[9] Hedge fund cost survey. https://www.altassets.net/pdfs/KPMG%20Hedge%20Fund%20Survey_Sept2008.pdf. Accessed: 2016-08-26.

[10] Maker market. https://github.com/makerdao/maker-market. Accessed: 2016-08-29.

[11] String technology. http://string.technology/. Accessed: 2016-08-26.

[12] t0. https://t0.com/home. Accessed: 2016-08-26.

[13] N. M. et al. The Dai Credit System. https://makerdao.com/dai02.pdf/.

[14] P. Wei, Y. Altshuler, and A. Pentland. Decoding Social Influence and the Wisdom of the Crowd in Financial Trading Network. *MIT Media Lab*.

[15] D. G. Wood. Ethereum: A Secure Decentralised Generalised Transaction Ledger. http://gavwood.com/paper.pdf/.

Appendix A. Terminology

Token: By the term token is meant, a digital token of value adhering to the Ethereum token standard [5].

Portfolio: A collection of smart-contracts, divided into a core smart-contract(s) and into auxiliary smart-contracts called modules.

Portfolio Manager: Portfolios are managed by one person or a group of persons, referred to as a Portfolio Manager.

Module: A module is on or a set of smart-contracts which has an auxiliary functionality to the core smart-contract of the portfolio.

Time Step: The term time step means, the time interval between blocks on the Ethereum Blockchain. Where time is taken from the block timestamp issued by the miner. For example $(t_i, t_{i+1}]$ is the time between after block with timestamp t_i and the following block with timestamp t_{i+1}.

[5]One example for a portal is the Melon portal: `https://melonport.com`.

Appendix B. Portfolio

Formally we expand the portfolio as follows. Assuming there are $n \in \mathbb{N}$ digital assets available. This constitutes the following vector set $\underline{a}$ of assets:

$$\underline{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \tag{5}$$

where a_k is the k-th available asset, for $k \in \mathbb{N}$. By convention the first asset a_1 represents Ether.

Then the *portfolio* $\underline{h}_m^{t_i}$, is defined as the vector set of asset holdings of a portfolio m at time t_i.

$$\underline{h}_m^{t_i} = \begin{pmatrix} h_{a_1}^{t_i} \\ \vdots \\ h_{a_n}^{t_i} \end{pmatrix} \in \mathbb{R}^n_{\geq 0} \tag{6}$$

where $h_{a_k}^{t_i}$ is the amount, in token units of a_k, a portfolio m holds at time t_i, for $k \in \mathbb{N}$.

Appendix C. Gross Asset Value

Let $\underline{p}^{t_i}$ be the vector set of asset prices, at time t_i. Then $\underline{p}^{t_i}$ is:

$$\underline{p}^{t_i} = \begin{pmatrix} p_{a_1}^{t_i} \\ \vdots \\ p_{a_n}^{t_i} \end{pmatrix} \in \mathbb{R}^n_{\geq 0} \tag{7}$$

where $p_{a_k}^{t_i}$ is the price per token unit of asset a_k in Ether, at time t_i, for $k \in \mathbb{N}$.

Note, since by convention a_1 represents Ether, and the prices are given in Ether the first price $p_{a_1}^{t_i}$ is always equal to one.

The Gross Asset Value or *GAV* $\hat{v}_{h_m}^{t_i}$ in Ether of portfolio $\underline{h}_m^{t_i}$ at time t_i is:

$$\hat{v}_{h_m}^{t_i} = \langle \underline{p}^{t_i}, \underline{h}_m^{t_i} \rangle = \sum_{k=1}^{n} p_{a_k}^{t_i} h_{a_k}^{t_i} \tag{8}$$

with the standard scalar product on $\mathbb{R}^n$. The GAV can be seen as the gross *value* of the portfolio.

Appendix D. Net Asset Value

The Net Asset Value or *NAV* $v_{h_m}^{t_i}$ in Ether of portfolio $\underline{h}_m^{t_i}$ at time t_i is:

$$v_{h_m}^{t_i} = \hat{v}_{h_m}^{t_i} - \textit{Management Fees}^{t_i} - \textit{Performance Fees}^{t_i} \tag{9}$$

Management Fees^{t_i} resp. *Performance Fees*^{t_i} is the management resp. performance fees given to the Portfolio Manager for timestep t_i.

Appendix E. Delta

To define the *Delta* $\Delta_{(t_i,t_j]}$ of a portfolio m, within the time $(t_i, t_j]$, where $t_i < t_j$, we first define the Delta of $\Delta_{(t_i,t_{i+1}]}$, i.e. the Delta of a single time step. Let be:

$$\begin{aligned} t_0 &= \textit{time of contract creation} \\ t_l &= \textit{time of first investment} \\ &= \min_{t_k \in [t_0, t_i]} \{v_{h_m}^{t_k} \neq 0\} \\ I^{t_i} &= \textit{Sum of all investments within } (t_{i-1}, t_i] \\ W^{t_i} &= \textit{Sum of all withdrawals within } (t_{i-1}, t_i] \end{aligned}$$

Where both I^{t_i} and W^{t_i} is a value in Ether. Then the Delta of a single time step is:

$$\Delta_{(t_i,t_{i+1}]} = \frac{v_{h_m}^{t_{i+1}} - I^{t_{i+1}} + W^{t_{i+1}}}{v_{h_m}^{t_i}} \tag{10}$$

By design, the Delta of a portfolio, is independent of funds invested or withdrawn within the time $(t_i, t_{i+1}]$. By factoring together these Deltas of single time steps we get the general definition of the Delta $\Delta_{(t_i,t_j]}$ of a portfolio m as:

$$\Delta_{(t_i,t_j]} = \begin{cases} 1 & \text{if} \quad t_j \leq t_l \\ \prod_{k=l}^{j-1} \Delta_{(t_k,t_{k+1}]} & \text{if} \quad t_i < t_l < t_j \wedge v_{h_m}^{t_k} \neq 0,\ k \in \{l, l+1, \ldots, j-1\} \\ \prod_{k=i}^{j-1} \Delta_{(t_k,t_{k+1}]} & \text{if} \quad t_l \leq t_i \wedge v_{h_m}^{t_k} \neq 0,\ k \in \{i, i+1, \ldots, j-1\} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Note, the case where $t_i < t_l < t_j$ and $v_{h_m}^{t_k} = 0$, for a $k \in \{l, l+1, \ldots, j-1\}$ is when the first investment has been made but the funds have been withdrawn completely at some point within time $(t_l, t_{j-1}]$. The GAV in this case can not be calculated by factoring together Deltas of single time steps, as this would mean a division through zero. The Delta in this case is set to 0 for all times, even if the portfolio receives investments in the future. The same is true for the second similar case where $t_l \leq t_i$ and $v_{h_m}^{t_k} = 0$, for a $k \in \{i, i+1, \ldots, j-1\}$.

By induction, we can see that the Delta $\Delta_{(t_i,t_j]}$ remains independent of funds invested or withdrawn within the time $(t_i, t_j]$.

Appendix F. Net Asset Value per Share

Expressed mathematically the NAV per share $p_m^{t_i}$ of portfolio m, at time t_i is:

$$p_m^{t_i} = \Delta_{(t_0,t_i]} \tag{12}$$

where t_0 is the time of contract creation. The price $p_m^{t_i}$ is denominated in Ether.

Appendix G. Share Price

The share price is defined by the Net Asset Value per Share (see appendix F) and is denominated in Ether.