# Table of Contents

## Thesis

[Cover](#_bookmark0) [Abstract](#_bookmark1) [Introduction](#_bookmark2)

[Problem Statement](#_bookmark3) [Method](#_bookmark4)

[Document Organisation](#_bookmark5) [Background](#_bookmark6)

[Current State of Fund Reporting](#_bookmark7) [Analysis](#_bookmark8)

[Use Cases](#_bookmark9)

[Data Classification](#_bookmark10) [Available Data](#_bookmark11)

[Solution](#_bookmark12)

[Specification](#_bookmark13) [System Overview](#_bookmark14)

[Software Architecture](#_bookmark15) [Screen/Paper Design](#_bookmark16) [Fund Report Data Schema](#_bookmark17)

Melon Auditing Contract Standard [Auditing Contract](#_bookmark18)

[Deterministic Data Extraction](#_bookmark19) [Standard Report Web Interface](#_bookmark20)

[Conclusion](#_bookmark21)

[Findings](#_bookmark22) [Future work](#_bookmark23) [Reflection](#_bookmark24)

## Appendices

[Technical Quickstart](#_bookmark25) [Project Management](#_bookmark26)

[Planning](#_bookmark27) [Milestones](#_bookmark28) [Journal](#_bookmark29) [Risks](#_bookmark30)

1.1

1.2

1.3

1.3.1

1.3.2

1.3.3

1.4

1.4.1

1.5

1.5.1

1.5.2

1.5.3

1.6

1.6.1

1.6.2

1.6.3

1.6.4

1.6.5

1.6.6

1.6.7

1.6.8

1.6.9

1.7

1.7.1

1.7.2

1.7.3

2.1

2.2

2.2.1

2.2.2

2.2.3

2.2.4

[Stakeholders](#_bookmark31) [Decisions](#_bookmark32)

[Iterations](#_bookmark33)

[Prototype](#_bookmark34)

[Goals](#_bookmark35) [Implementation](#_bookmark36) [Evaluation](#_bookmark37)

[Internal](#_bookmark38)

[Stakeholder Feedback](#_bookmark39) [Explorative User Testings](#_bookmark40)

[Conclusion](#_bookmark41) [First draft](#_bookmark42)

[Goals](#_bookmark43) [Implementation](#_bookmark44)

[Validative User Testing](#_bookmark45) [Auditing Contract](#_bookmark46)

[Final product](#_bookmark47)

[Changes to first draft](#_bookmark48) [Research](#_bookmark49)

[Key Investor Information Document](#_bookmark50) [Timely reports](#_bookmark51)

[MiFID II and PRIIP](#_bookmark52)

[Melon Risk Engineering](#_bookmark53) [uPort](#_bookmark54)

[Solidity](#_bookmark55)

[Solidity Cheatsheet](#_bookmark56) [Data Classification](#_bookmark57) [Call for Collaborators](#_bookmark58)

[Minutes](#_bookmark59) [References](#_bookmark60) [Glossary](#_bookmark61)

[Original Project Submission](#_bookmark83) [Declaration of Academic Integrity](#_bookmark84) [List of figures](#_bookmark85)

[List of tables](#_bookmark86)

2.2.5

2.2.6

2.3

2.3.1

2.3.1.1

2.3.1.2

2.3.1.3

2.3.1.3.1

2.3.1.3.2

2.3.1.3.3

2.3.1.4

2.3.2

2.3.2.1

2.3.2.2

2.3.2.3

2.3.2.4

2.3.3

2.3.3.1

2.4

2.4.1

2.4.2

2.4.3

2.4.4

2.4.5

2.4.6

2.4.7

2.4.8

2.4.9

2.5

2.6

2.7

2.8

2.9

2.10

2.11

# Melon Reporting and Auditing

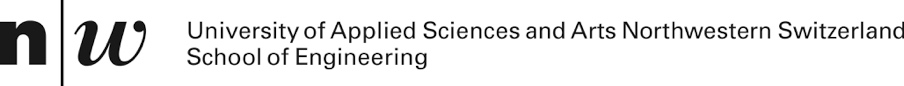


Bachelorthesis by Simon Emanuel Schmid and Benjamin Zumbrunn Spring Semester 2018

University of Applied Sciences Northwestern Switzerland FHNW, School of Engineering. BSc Computer Sciences / iCompetence.

Coaches: Prof. Dr. Sarah Hauser, Markus Knecht Customer: Melonport AG

Brugg, August 16th 2018



# Abstract

Melon is a fully decentralized asset management protocol which allows anyone to set up, manage and/or invest in an investment fund of digital assets in a secure, robust and permissionless manner. In three words: decentralized investment infrastructure.

From [5]

Previous to this thesis, managers, investors, auditors and regulators of Melon funds did not have a reasonable tool to analyze how a fund is managed and how it performs. The *Reporting and Auditing* module resolves this highly demanded part of Melon.

There are four crucial stakeholders, each with their own requirements on Reporting and Auditing:

Fund managers are interested on performance and want to have hints on how to improve their trades Investors are interested in past management of a fund

Auditors need to verify the integrity of the manager's work Regulators need to verify these audits

The final product contains the following parts:

The *Report Data Extractor* extracts all needed data to visualize a fund over a given timespan.

The *Standard Report Web Interface* visualizes this data in the form of a report which is of use for all stakeholders. The *Auditing Web App* provides auditors with the ability to pass an opinion of a fund on a specific timespan which then can be verified by regulators.

Through successful explorative and validative user testings, the solution proves to be an effective tool which pleases all stakeholders.

# Introduction

## Organization

The main entry point for this project is the open source Github repository [github.com/melonproject/reporting](https://github.com/melonproject/reporting). The documentation can be found on Gitbooks: <https://melon-reporting.now.sh/>.

Project management is done with a simple Kanban board as a Github Project: [github.com/melonproject/reporting/projects/1](https://github.com/melonproject/reporting/projects/1)

Single tasks are managed as Github Issues: [github.com/melonproject/reporting/issues](https://github.com/melonproject/reporting/issues)

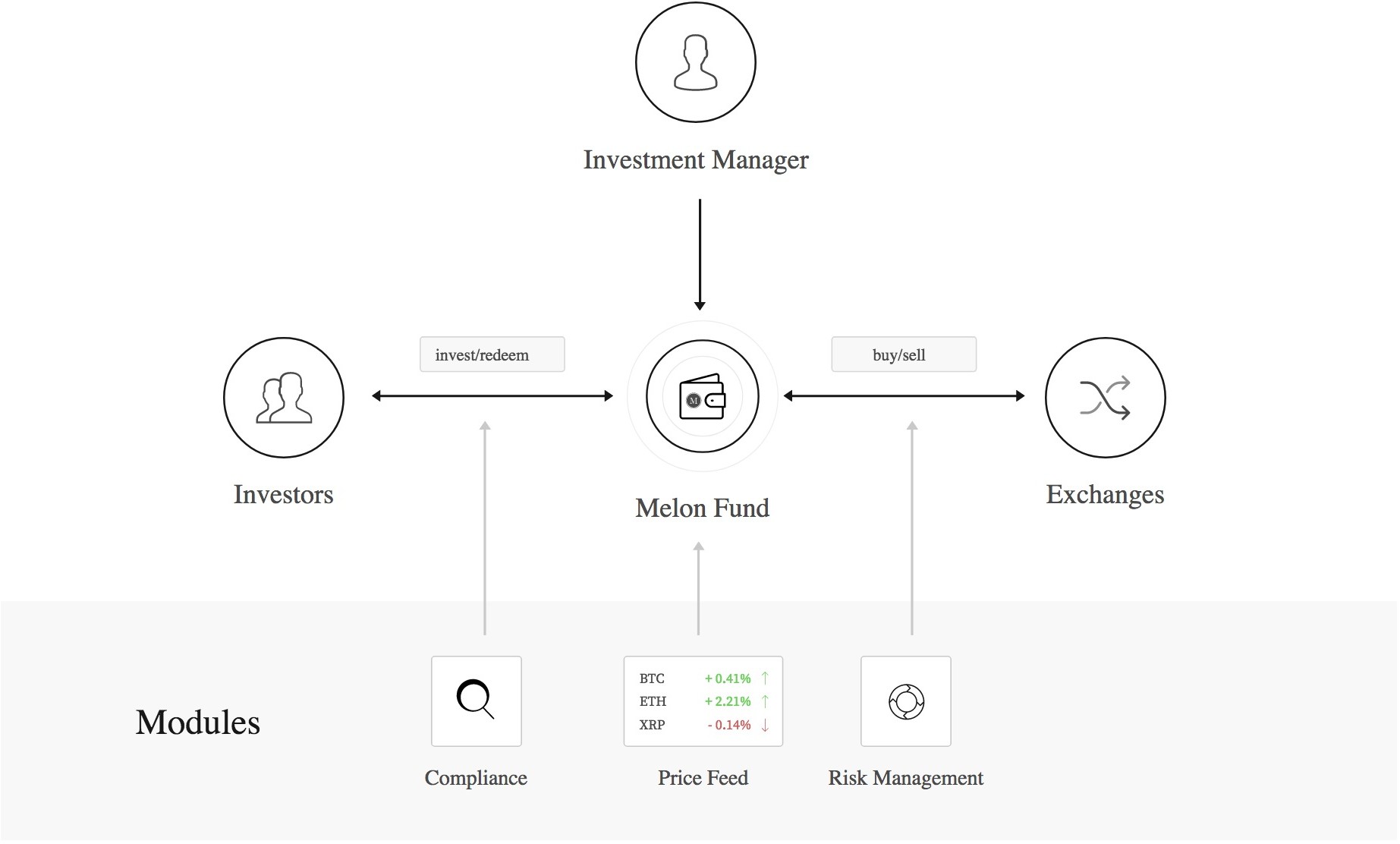
# Problem statement

## Melonport AG

Melonport AG is a company with the mission to build the Melon Protocol until February 2019 and then hand it over to a self governing community.

## Melon Protocol

Melon is a fully decentralized asset management protocol which allows anyone to set up, manage and/or invest in an investment fund of digital assets in a secure, robust and permissionless manner. In three words: decentralized investment infrastructure.



*Image & text by Melonport AG*

Technically it is a set of [Solidity](#_bookmark81) smart contracts running on the Ethereum Blockchain. There also exists a frontend acting as reference implementation on how to interact with these smart contracts.

## Vision

Creating functionality on top of the Melon protocol that automates reporting/auditing almost completely:

1. Something that a real fund manager would be able to confidently say: “This solves my reporting issues and makes my life a lot easier”
2. Something that can be show-cased to [FINMA](#_bookmark72) (and other regulators) and show them how: "This will make

*their* life over-seeing a lot easier”

## Hypothesis

It is possible to extract and visualize all relevant data from the Melon protocol on the Ethereum blockchain in a way that could be legally acceptable by regulators. Furthermore, this data can be audited and digitally signed and a track record of these audits can be placed on the blockchain again.

## Boundaries

### Legal

This is a technical thesis and therefore we do not deeply research into the legal aspects of fund management and reporting. But we will find ways how technology can support the legal processes.

### Traditional Auditing

This thesis focuses on the possibilities of auditing of funds running on the blockchain. This also implies the immutability of on-chain data and the blockchain as the single source of truth.

Furthermore do we not include any aspect of other traditional financial auditing such as: Third-party risks, company structure audits, internal control policies, banking reconciliation (not necessary), taxes and others.

That said, an auditor can still perform these audits and place its signed opinion on-chain.

### Technical

**Melon Fund System**

A lot of functionality of funds depends on third party modules: Price feeds, [participation](#_bookmark77), exchanges. In this thesis, we only guarantee the official modules provided by Melonport. If modules outside of this scope could provide important functionality, we document this, but do not implement the necessary bridges.

We try to adapt to the latest Melon smart contracts but fall back to the last known working ones if the latest causing too much problem. Last known working version is [v0.7.0](https://github.com/melonproject/smart-contracts/tree/v0.7.0).

### On-chain data

We only work actively with data that is available on-chain. Off-chain data could be merged into the reporting system and we document the interfaces to do so as future work.

### Other

We exclude margin-trading in general for this thesis.

## Approach

Melonport and most blockchain based companies see themselves as innovating startups trying to build a new and better world. Therefore it does not see itself as a service provider and its possible users as customers. The development process is more like an open community where input is very welcome but also critically assessed.

“If I had asked people what they wanted, they would have said faster horses.” *Henry Ford*

# Method

Due to the nature of tight time constraints in this thesis we opt for a mixed approach between iterative and waterfall project management:

Plan 3 iterations

Every iteration shall have deliverables which we can use to collect feedback.

Distribute feedback meetings with different stakeholders: Every 3-4 weeks with the coaches

After each iteration with the client & coaches

The detailed project management & iteration planning is in the appendix:

Project Management Iterations

Here is a short overview over the iterations and deliverables:

## Prototype

Build something fast & simple to show to various stakeholders and gather feedback.

## First draft

Having a functional end to end prototype ready that touches all important parts.

## Final product

Iterate over the first draft to bring it to the final product that is submitted.

# Document organization

In the chapter *Background* we have a closer look on how fund reporting is done traditionally.

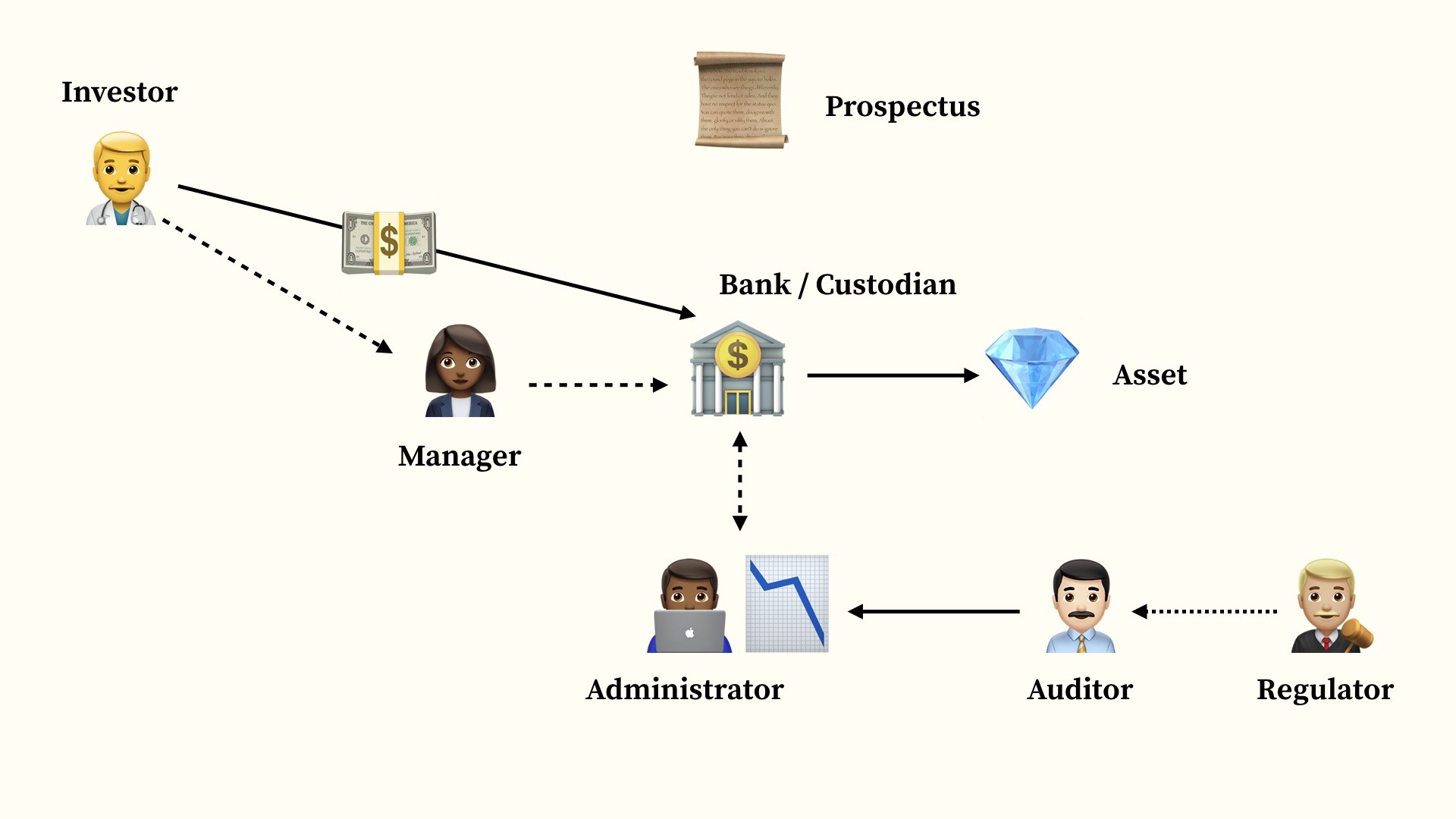
In the chapter *Analyze* we dissect what is expected from the *Auditing and Reporting* module from the perspective of different stakeholders. Furthermore, we analyze the data that the Melon protocol provides and classify it.

In the chapter *Solution* we dive into the architecture, design and implementation of all submodules.

In the chapter *Conclusion* we reflect on the solution, summarize findings we achieved by the thesis and propose a list of future work on *Auditing and Reporting* in Melon.

# Background

**Current State of Fund Reporting**



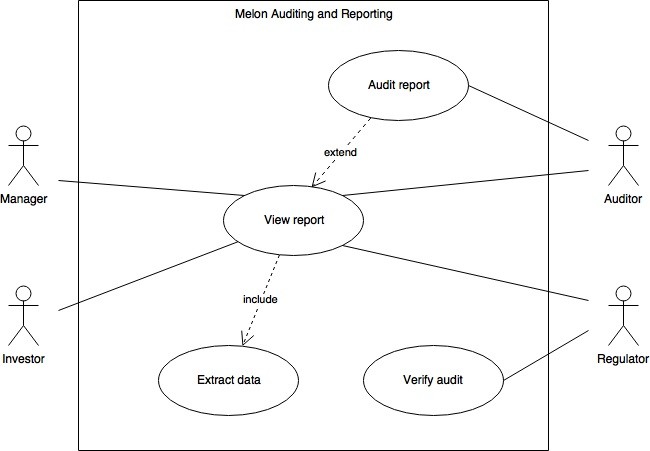
The traditional model of fund reporting and auditing is a complex network of different companies and roles that have to supervise each other in order to protect the investors and make it difficult to launder money or violate the law otherwise.

This makes it expensive and cumbersome, data is not available or different parties claim to have different data.

# Analysis

**Use Cases**

## Use case diagram



**Actors**

Investor: John Orthwein Manager: Mona El Isa Auditor: tbd??

Regulator: tbd??

### Investor

Investors want a report to be presented to them as quickly as possible. They care about the performance of a fund since its [inception](#_bookmark73). They are also interested if a fund is regularly audited. This information contributes to the decision whether an investor is willing to invest in a fund or not. Investors also want to rely on risk modules by having the option to forbid new trades if funds are not regularly or properly audited.

### Manager

Managers may want to analyze funds in a more specific scope so they can adjust their trading approaches. They are also interested in comparing their fund to others.

### Auditor

Auditors rely heavily on the integrity of the presented data as they are the ones to sign it. They want to set an opinion on a specific timespan after revieving the data. They are interested in past audits of a fund so they know where additional audits are needed.

### Regulator

Regulators need a way to verify if a fund is regularly audited and the signed data of an audit is valid.

## Use case description

### Extract Data

Actors can extract report data of a fund on the timespan they provide.

### View Report

Actors can view a visual representation of the extracted data with a report.

### Audit Report

Actors can add an audit on a fund over a timespan with a comment in form of an opinion.

### Verify audit

Actors can verify an audit by comparing the hash of the audit and the hash of their created report.

# Data classification

For data classification, we studied the [CAP](#_bookmark65) theorem and the properties of [ACID](#_bookmark63). As all of our data for the *Reporting and Auditing* module will be fetched from the Ethereum blockchain, these paradigms do not perfectly apply to our use case. Therefore, we came up with our own classification which we qualify here.

Further reasoning on this can be found in the [Appendices](#_bookmark57).

## Classifications

### Data Type

**Short text**: A name, some key words or a character.

**Long text**: A long text, usually written in prose.

**Number**: Any representation of a number, e.g. a price or a percentage.

**Address**: Data that is considered to be an address, e.g. an Ethereum address.

**Time**: Any format representing time. This is mostly a unix timestamp, so the accuracy is at most one second.

**Container**: A more complex type, holding multiple values.

### Source

**On-chain call**: Data can be retrieved from the blockchain directly by calling a function. *Examples*: current price, fund name, recent trades, ...

**On-chain event**: Data can be retrieved from the blockchain by searching for events. This data cannot be used by smart contracts. I.e. a smart contract cannot check if a certain event happened. *Examples*: Price history

**Off-chain**: Data that is not retrievable from the blockchain.

**Free text**: General text. Usually lawyer jargon.

**Given**: Data that is given by a template/recommendation, or is known through other sources.

### Availability

**Simple**: Data is directly available through a simple (API-) call. *Example*: Fund name, [AUM](#_bookmark64), asset prices, orderbook ...

**Complex**: Data is generally available but we need to build special tools to extract it. *Examples*: Historical prices

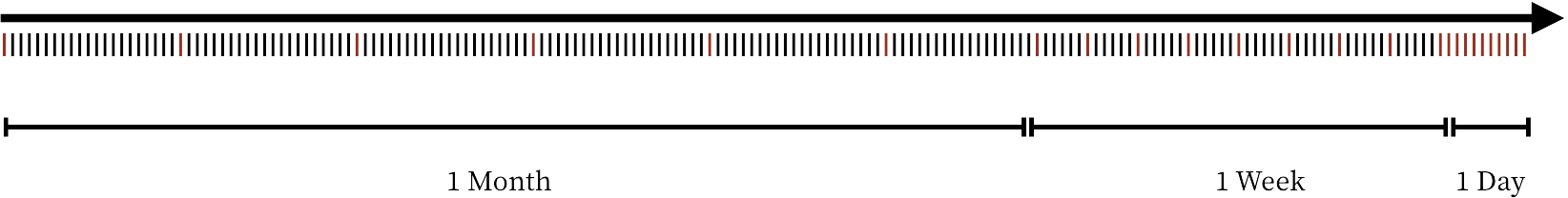
**N/A**: Data is not available (yet)

### Consistency

**Static**: Once written this data does not change anymore. *Examples*: Fund name, ... **Configuration**: Data that is configured out of a predefined set of possible options. **Generated**: Data that is generated automatically. *Examples*: Address, indexes, ... **Live**: Data that can change frequently. *Examples*: Prices, orderbook, [AUM](#_bookmark64), ...

**Archive**: List data that stays once written. Archive data can be consolidated for display purposes but needs to be preserved. *Examples*: Trade history, [participation](#_bookmark77) history, ...

**Historic**: List data that stays once written but it's importance/density lessens with age: *Examples*: Price history; It is important to know the exact price for every minute for the last 24h but older prices can be stored in less density, e.g every hour. The older the data, the lesser the density.



*Red lines: saved data, black lines: available data*

## Data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data** | **Data Type** | **Source** | **Availability** | **Consistency** |
| Fund name | Short text | On-chain call | Simple | Static |
| Fund address | Address | Given | Simple | Generated |
| Timespan start | Time | Given | Simple | Configuration |
| Timespan end | Time | Given | Simple | Configuration |
| Manager | Address | On-chain call | Simple | Static |
| [Inception](#_bookmark73) | Time | On-chain call | Simple | Generated |
| Quote symbol | Short text | On-chain call | Simple | Static |
| Quote address | Address | On-chain call | Simple | Static |
| Exchange address | Address | On-chain call | Simple | Static |
| Exchange name | Short text | On-chain call | Simple | Static |
| Total supply | Number | On-chain call | Simple | Live |
| [Participation](#_bookmark77) | Container | On-chain event | Complex | Archive |
| Audit | Container | On-chain call | Simple | Archive |
| Holding symbol | Short text | On-chain call | Simple | Static |
| Holding address | Address | On-chain call | Simple | Static |
| Holding balance | Number | On-chain call | Simple | Live |
| Price history | Number | On-chain event | Complex | Archive |

**Reasoning**

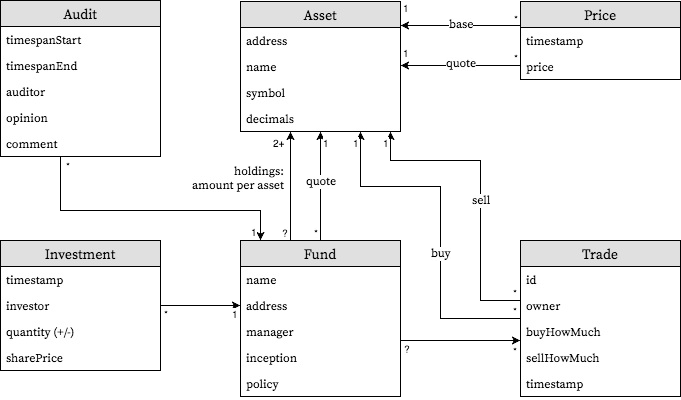
The reasoning behind this classification is described in Appendix D.

# Available Data

## On-Chain Data

The following [ERM](#_bookmark69) is a representation of the minimalistic data available on the blockchain of a fund to build charts and reports. All data can be derived from this model.

Please note:

The data is not actually stored like this on the blockchain, this is just for better understanding The Audit entity is added by this thesis

**Quantifiers** (same as regex):

1: Only one

\*: Zero or more

+: One or more

?: Zero or one 2+: Two or more

### Explanation

All data is considered immutable. Example: Once a fund is setup with certain parameters, it cannot change anymore.

**Fund**: The fund is in the center and the starting point.

Every fund has one quote asset in which its value is denominated. Usually ETH or [DAI](#_bookmark66).

**Investment**: In the beginning one or more investors invest in the fund. Usually with the quote asset. Example of first investment: Buying 1 share buy putting one ETH in the fund. --> Shareprice of 1.

**Asset**: Basically a fund holds different assets, one of them the quote asset. Example: The fund holds 2 ETH and 3 MLN.

**Price**: Every trading pair (base asset / quote asset) has a recent price and a price history. Example: The price of MLN/ETH is 0.05 and was 0.06 two weeks ago.

**Trade**: The fund can exchange those assets through trades. Example: Selling 20 MLN for 1 ETH.

**Audit**: This thesis adds the audit functionality on top of the fund.

An audit is a timespan with an opinion and an optional comment.

Example: PwC audited the fund with address 0x123 for the timespan from 1. January 2018 to 31. March 2018 with a "unqualified opinion" and a comment of "all good".

## Off-Chain Data

The data extractor shall always produce the report on same input arguments. An extractor function must therefore be deterministic. With off-chain data, determinism is not given naturally like with on-chain data.

There might be a way to include off-chain data into the current architecture nontheless, if this will be of need in the future.

One could hash off-chain data and put this hash on the chain as an anchor. This anchor could then be used to resolve the data.

But for this thesis we do not include off-chain data.

# Solution

**Specification**

*Note:* The following specification serves as the official project scope agreed upon by the coaches and the client during the prototype presentation meeting on 3rd Mai 2018 in Zug.

## Stakeholder needs

### Investor

Timespan: Mainly [inception](#_bookmark73) to now Comparable and complete fund reports

Is a fund regularly audited (which is also backed by risk modules) Key information:

Share price & history Assets under management Tokens & allocation

Audit interval & coverage Auditors

Manager Legal entity Fees

### Manager

Timespans:

[Inception](#_bookmark73) to now

Monthly, quarterly, yearly, year to date (YTD) Custom

Compare own fund to others Auditing status

Key information: All of investor plus: [Participation](#_bookmark77) allocation & history Detailed trades history and analysis

### Auditor

All of manager plus:

Inspect & validate data especially trade history Sign & add a report

### Regulator

Check if a fund is regularly audited Perform spot checks of audits

Is the signed data of an audit valid

## Detailed use case description

### Extract Data

Extract report data on-chain.

When data is extracted with the same arguments, the datahash will always be the same.

Input arguments are:

Fund (address)

Timespan (timestamp to timestamp) The system extracts following data:

General: Fund name, share price, policies, etc. Holdings: List of tokens with quantity, price (history), ... Trades: List of trades in the defined timespan

[Participation](#_bookmark77): List of invests/redeem in the defined timespan Audits: List of audits in the defined timespan

### View Report

All actors (Auditor, Manager, Investor, Regulator) can view reports. Create a report out of the fund data and display it to the user.

This rendering is complete and comparable.

Complete: All underlying report data is visually represented

Comparable: For different funds with the same timespan the rendered reports look similar.

*Optional*: Make the report printable

### Audit Report

When an auditor has reviewed a report, he can create an audit on the blockchain. Input arguments are:

Fund (address) Datahash (of report data)

Timespan (timestamp to timestamp) Opinion (enum)

Comment (string)

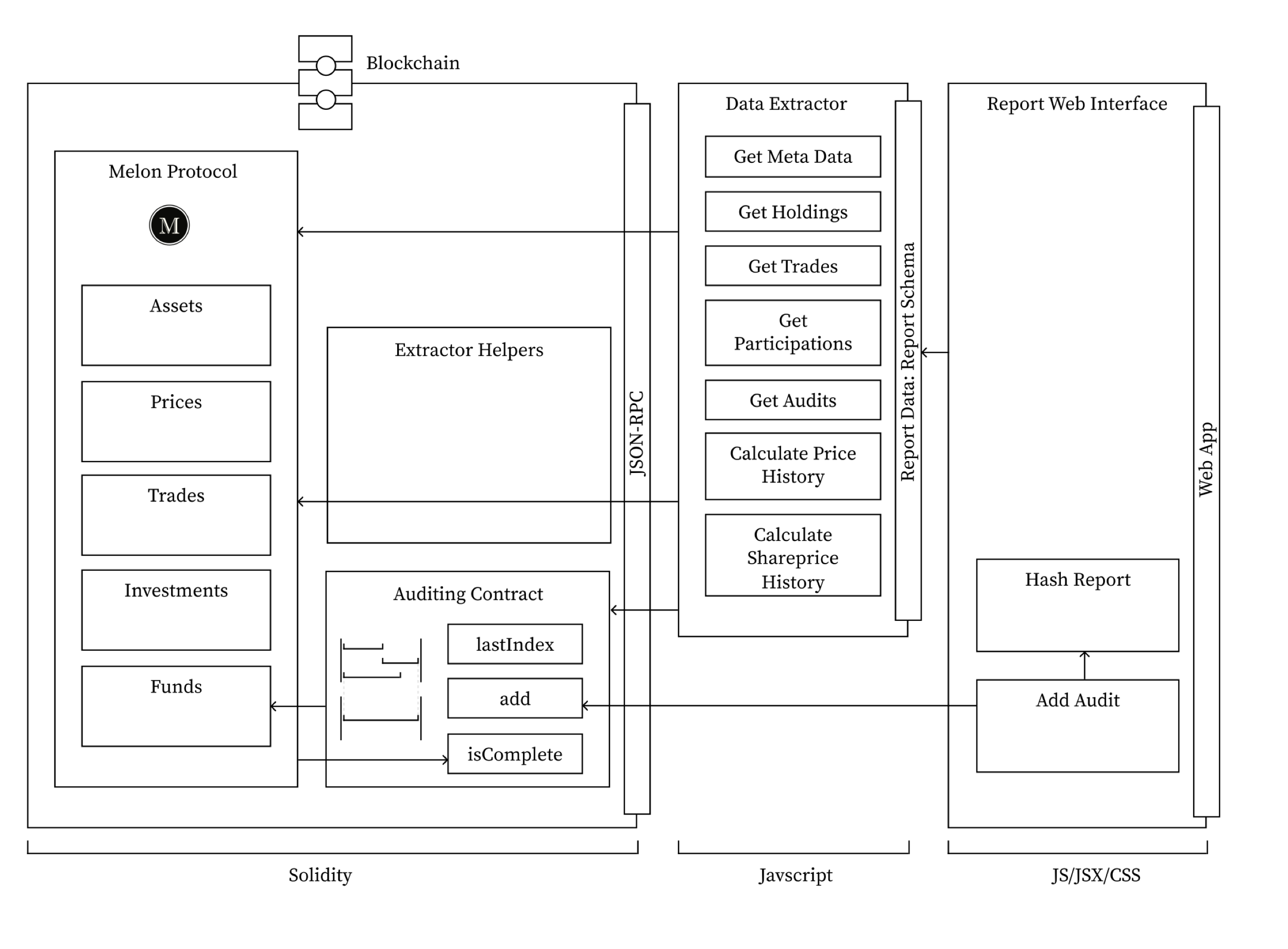
### Verify audit

A regulator can verify an audit. This is basically the ability to make a spot check of audits:

Recreate report according to fund address and timespan Check if datahash matches

Validate data. Although if the auditor trusts its report generation algorithm and datahash matches, than the report is considered valid.

# System Overview

****

## Explanation

At highest level the system can be divided in blockchain and non-blockchain parts

The non-blockchain parts communicate with the blockchain through the Ethereum JSON RPC protocol [6] Code that runs on the blockchain is written in [Solidity](#_bookmark81)

Code that interacts with the blockchain is written in JavaScript/React

At the core of the system is the Melon Protocol, which is provided by Melonport The Auditing Contracts adds the auditing functionality on top of the Melon Protocol

In the future the Melon Protocol risk management module might use the Auditing Contract to permit/forbit trades wether the fund is properly audited or not.

The data extractor can extract data from the blockchain in the specified Report Data Schema format. It has different strategies to do so:

Directly call a function of the Melon Protocol (easy and fast)

Search the logs for events of the Melon Protocol (cumbersome and slow) Use a on-chain data extractor helper (fast but troublesome to set up) Directly call a function of the Auditing Contract (easy and fast)

The Report Web Interface uses this extracted data to render a standard report as a web app.

This web app has some interactive features such as selecting the timespan but is fairly static to also be usable on paper.

Also this web app has the functionality to add an audit over a given timespan and put this back on the blockchain.

Cryptographically speaking: The report data for the given timespan is hashed and put on the blockchain through a transaction signed by the auditor.

# Software Architecture

## General considerations

Since the stability of the Melon Protocol is not given yet the whole system is modularized into modules which are fully functional without dependencies.

## Data Extraction

Parity API...

## Data Schema

### JSON Schema

Validation of the report data is done using [JSON Schema](http://json-schema.org/). This adds an extra layer of integrity, so an auditor is able to verify that the hash they will sign is compliant with the *Fund Report Data Schema*.

## Auditing

### web3.js

[web3js](https://web3js.readthedocs.io/) gives us the ability to call functions on our auditing smart contract.

### Forge

With the npm package [Forge](https://github.com/digitalbazaar/forge) we hash the report data.

### MetaMask

[MetaMask](https://metamask.io/) lets us connect to an Ethereum chain (Kovan, Mainnet etc.) and sign the auditing data without running a full Ethereum node by ourselves.

## Smart contracts

### dapp tools

With [dapp tools](https://dapp.tools/) we used a toolsuite for developing Ethereum smart contracts that gave us the ability to automate the following:

compile smart contracts (with solc) debug smart contracts (with Hevm) test smart constracts (with ds-test) extract an [ABI](#_bookmark62) (with solc)

### Architecture

The main smart contract for auditing is *Auditing.sol*.

With the interface *AuditingInterface.sol*, we follow a modular apprach. Users of the Melon protocol are encouraged to implement their own Auditing contract if they are not pleased with our solution. A possible change on our contract might be that auditor whitelisting will be approached differently, e.g. by opening the *add* function (hence the ability to add an audit on a fund) to all addresses.

Example: contract MyAuditing is AuditingInterface { ... }

When they follow the specification of the auditing interface, other implementations are guaranteed that their contract will comply with the rest of the Reporting and Auditing architecture.

The [ABI](#_bookmark62) of the auditing smart contract provides an interface to the Auditing functionality of the *Report Web Interface*.

### Unit tests Functional tests

File under test: Auditing.sol

Test file: Auditing.t..sol

TODO graphs of implemented tests (some of the timelines...)

### "Bulk" tests

Files under test: Auditing.sol, AuditingFirst.sol

Test files: AuditingBulkOne.t.sol, AuditingBulkTen.t.sol, AuditingBulkHundred.t.sol

With these unit tests, we were able to compare two variants of the Auditing Contract implementation. We take a closer look at the results in [this chapter](#_bookmark18)

## Web Interface

Built on top of React and Next.js to enable server side rendering (similar to the setup of the Melon Manager Interface)

Separate development environment for components with <https://storybook.js.org/>

# Screen/Paper Design

The vision of the design is to be adaptive: That the same design looks familiar on a printed paper to correspondent with traditional reporting mind-set. But also to have subtle interactions in the browser that make its usage more efficient.

Download mockup PDF

# Fund Report Data Schema

## Description

We need a way to define the data from which a report can be produced for viewing and auditing. The [JSON Schema](http://json-schema.org/) is a good fit, because we can specify types, involved data structures, constraints and describe the data directly.

The Schema *Fund Report* is a JSON to define our data JSON files (acutal instances of Fund Report data), basically a meta-JSON.

# Schema "Fund Report"

// TODO:

// add descriptions where useful

// $id when schema place is fixed --> link to schema on github

// some bigNumbers might just be normal javascript numbers

{

"$schema": "[http://json-schema.org/draft-07/schema#](http://json-schema.org/draft-07/schema)", "title": "Fund Report",

"$version": "0.1",

"description": "Fund data to generate a report from and create a hash for auditing.", "type": "object",

"definitions": { "nonNegativeInteger": {

"type": "integer", "minimum": 0

},

"ethereumAddress": { "type": "string",

"pattern": "^0x(\\d|[A-F]|[a-f]){40}$"

},

"erc20TokenSymbol": { "type": "string", "pattern": "^([A-Z]){1,9}$"

},

"bigNumber": { "type": "string",

"pattern": "^\\d\*.\\d\*$"

},

"timePeriod": { "type": "string", "enum": [

"day",

"month", "year"

]

},

"token": {

"type": "object", "properties": {

"symbol": {

"$ref": "#/definitions/erc20TokenSymbol"

},

"address": {

"$ref": "#/definitions/ethereumAddress"

}

},

"required": [

"symbol", "address"

]

},

"exchange": { "type": "object", "properties": {

"id": {

"type": "string"

},

"address": {

"$ref": "#/definitions/ethereumAddress"

}

},

"required": [

"id", "address"

]

},

"orderSide": { "type": "object", "properties": {

"token": {

"$ref": "#/definitions/token"

},

"howMuch": {

"$ref": "#/definitions/bigNumber"

}

},

"required": [

"token", "howMuch"

]

},

"participationType": { "type": "string", "enum": [

"invest", "redeem"

]

}

},

"properties": { "meta": {

"type": "object", "properties": {

"fundName": {

"type": "string"

},

"timespanStart": {

"$ref": "#/definitions/nonNegativeInteger"

},

"timespanEnd": {

"$ref": "#/definitions/nonNegativeInteger"

},

"fundAddress": {

"$ref": "#/definitions/ethereumAddress"

},

"inception": {

"$ref": "#/definitions/nonNegativeInteger"

},

"quoteToken": { "type": "object", "properties": {

"symbol": {

"$ref": "#/definitions/erc20TokenSymbol"

},

"address": {

"$ref": "#/definitions/ethereumAddress"

}

},

"required": [

"symbol", "address"

]

},

"manager": {

"$ref": "#/definitions/ethereumAddress"

},

"exchanges": { "type": "array", "items": {

"$ref": "#/definitions/exchange"

}

},

"legalEntity": { "type": "array", "items": {

"type": "string"

}

},

"strategy": {

"type": "string"

},

"policy": { "type": "object", "properties": {

"portfolio": { "type": "object", "properties": {

"maxPositions": {

"$ref": "#/definitions/nonNegativeInteger"

},

"bestPriceTolerance": {

"$ref": "#/definitions/bigNumber"

},

"maxTrades": { "type": "object", "properties": {

"threshold": {

"$ref": "#/definitions/nonNegativeInteger"

},

"timePeriod": {

"$ref": "#/definitions/timePeriod"

}

},

"required": [ "threshold", "timePeriod"

]

},

"maxVolume": { "type": "object", "properties": {

"threshold": {

"$ref": "#/definitions/bigNumber"

},

"timePeriod": {

"$ref": "#/definitions/timePeriod"

}

},

"required": [ "threshold", "timePeriod"

]

},

"volatilityThreshold": {

"$ref": "#/definitions/bigNumber"

}

},

"required": [ "maxPositions", "bestPriceTolerance", "maxTrades", "maxVolume", "volatilityThreshold"

]

},

"tokens": { "type": "object", "properties": {

"whitelist": { "type": "array", "items": {

"$ref": "#/definitions/token"

}

},

"liquidityInDays": {

"$ref": "#/definitions/nonNegativeInteger"

},

"marketCapRange": { "type": "object", "properties": {

"min": {

"$ref": "#/definitions/nonNegativeInteger"

},

"max": {

"$ref": "#/definitions/nonNegativeInteger"

}

},

"required": [ "min"

]

},

"volatilityThreshold": {

"$ref": "#/definitions/bigNumber"

}

},

"required": [ "whitelist", "liquidityInDays", "marketCapRange", "volatilityThreshold"

]

},

"participation": { "type": "object", "properties": {

"name": {

"type": "string"

},

"address": {

"$ref": "#/definitions/ethereumAddress"

}

},

"required": [

"name", "address"

]

}

},

"required": [ "portfolio", "tokens", "participation"

]

}

},

"required": [

"fundName", "timespanStart", "timespanEnd", "fundAddress", "inception", "quoteToken", "manager", "exchanges", "policy"

]

},

"holdings": { "type": "array", "items": {

"type": "object", "properties": {

"token": {

"$ref": "#/definitions/token"

},

"quantity": {

"$ref": "#/definitions/bigNumber"

},

"priceHistory": { "type": "array", "items": {

"$ref": "#/definitions/bigNumber"

}

}

},

"required": [

"token", "quantity", "priceHistory"

]

}

},

"trades": { "type": "array", "items": {

"type": "object", "properties": {

"buy": {

"$ref": "#/definitions/orderSide"

},

"sell": {

"$ref": "#/definitions/orderSide"

},

"exchange": {

"$ref": "#/definitions/exchange"

},

"timestamp": {

"$ref": "#/definitions/nonNegativeInteger"

},

"transaction": {

"$ref": "#/definitions/ethereumAddress"

}

},

"required": [

"buy",

"sell", "exchange", "timestamp", "transaction"

]

}

},

"participations": { "type": "array", "items": {

"type": "object", "properties": {

"investor": {

"$ref": "#/definitions/ethereumAddress"

},

"token": {

"$ref": "#/definitions/token"

},

"type": {

"$ref": "#/definitions/participationType"

},

"amount": {

"$ref": "#/definitions/bigNumber"

},

"shares": {

"$ref": "#/definitions/bigNumber"

},

"timestamp": {

"$ref": "#/definitions/nonNegativeInteger"

}

},

"required": [ "investor", "token",

"type",

"amount",

"shares", "timestamp"

]

}

},

"audits": { "type": "array", "items": {

"type": "object", "properties": {

"auditor": {

"$ref": "#/definitions/ethereumAddress"

},

"dataHash": {

"type": "string"

},

"timespanStart": {

"$ref": "#/definitions/nonNegativeInteger"

},

"timespanEnd": {

"$ref": "#/definitions/nonNegativeInteger"

}

},

"required": [ "auditor", "dataHash", "timespanStart", "timespanEnd"

]

}

}

},

"required": [

"meta", "holdings", "trades", "participations", "audits"

]

}

# Example Data

TO DISCUSS:

maybe use schema built-in date-time type for readability?

{

"meta": {

"fundName": "Melon Crypto Capital", "timespanStart": 1524739040740,

"timespanEnd": 1524741632780,

"fundAddress": "0x9395de1F82233A1A19319966672E0cE6B32dF080",

"inception": 1524710528740, "quoteToken": {

"symbol": "DAI",

"address": "0x9A0Bd6b8c445D67277A8b4b4cEf2339d4b7C9772"

},

"manager": "0x54bb6bcbE88e3EcD2788380a686064984de78531", "exchanges": [

{

"id": "Radar Relay",

"address": "0x2A3430a4875D0440F1Dd80Cd1eB406f2E063E61f"

},

{

"id": "ERC Dex",

"address": "0x5d263eb2C080a48F64179689549AAf6056410dfB"

}

],

"legalEntity": [

"Melon Crypto Capital AG", "Herrengasse 18",

"FL-9490 Vaduz",

"Liechtenstein"

],

"strategy": "Melon Crypto Capital focuses on the most promising ...", "policy": {

"portfolio": { "maxPositions": 100,

"bestPriceTolerance": "0.05", "maxTrades": {

"threshold": 50, "timePeriod": "month"

},

"maxVolume": {

"threshold": "10000.0000", "timePeriod": "month"

},

"volatilityThreshold": "0.25"

},

"tokens": {

"whitelist": [

{

"symbol": "MLN",

"address": "0x153F602ad18BBD1546b674cFDBe05a8ba72A1d57"

},

{

"symbol": "ETH",

"address": "0xad0E75B07cb4b1004A96Fa9a8D6F5e0B4b4fdA16"

},

{

"symbol": "DAI",

"address": "0x9A0Bd6b8c445D67277A8b4b4cEf2339d4b7C9772"

}

],

"liquidityInDays": 10, "marketCapRange": {

"min": 10000000,

"max": 10000000000

},

"volatilityThreshold": "0.35"

},

"participation": {

"name": "Bitcoin Suisse KYC",

"address": "0x9Ca935A5be2f7eD83e453647dBa2179D2cfDa1D8"

}

}

},

"holdings": [

{

"token": { "symbol": "MLN",

"address": "0x153F602ad18BBD1546b674cFDBe05a8ba72A1d57"

},

"quantity": "5.4910000000000",

"priceHistory": [ "1.000000",

"1.506000",

"1.406860"

]

},

{

"token": { "symbol": "ETH",

"address": "0xad0E75B07cb4b1004A96Fa9a8D6F5e0B4b4fdA16"

},

"quantity": "0.2530000000000",

"priceHistory": [ "2.100123",

"2.306630",

"2.206153"

]

},

{

"token": { "symbol": "DAI",

"address": "0x9A0Bd6b8c445D67277A8b4b4cEf2339d4b7C9772"

},

"quantity": "106.3490000000000",

"priceHistory": [ "1.51198",

"1.69279",

"1.76345"

]

}

],

"trades": [

{

"buy": {

"token": { "symbol": "MLN",

"address": "0x153F602ad18BBD1546b674cFDBe05a8ba72A1d57"

},

"howMuch": "3.982323249"

},

"sell": {

"token": { "symbol": "DAI",

"address": "0x9A0Bd6b8c445D67277A8b4b4cEf2339d4b7C9772"

},

"howMuch": "3.982323249"

},

"exchange": {

"id": "Radar Relay",

"address": "0x2A3430a4875D0440F1Dd80Cd1eB406f2E063E61f"

},

"timestamp": 1524739030740,

"transaction": "0x76856aF5b24b29C8cDA09D8d27f527211747819c"

},

{

"buy": {

"token": { "symbol": "ETH",

"address": "0x153F602ad18BBD1546b674cFDBe05a8ba72A1d57"

},

"howMuch": "3.982323249"

},

"sell": {

"token": { "symbol": "DAI",

"address": "0x9A0Bd6b8c445D67277A8b4b4cEf2339d4b7C9772"

},

"howMuch": "2.2939423"

},

"exchange": {

"id": "Radar Relay",

"address": "0x2A3430a4875D0440F1Dd80Cd1eB406f2E063E61f"

},

"timestamp": 1524739030740,

"transaction": "0x76856aF5b24b29C8cDA09D8d27f527211747819c"

}

],

"participations": [

{

"investor": "0xcDcCB1259CF7388D9018009349C945Cc35d5AFbE",

"token": { "symbol": "DAI",

"address": "0x9A0Bd6b8c445D67277A8b4b4cEf2339d4b7C9772"

},

"type": "invest", "amount": "23478.8230000",

"shares": "2387.923990",

"timestamp": 1524737164199

},

{

"investor": "0xd1324AEd96fC94e219c3663A665c6e398D8634Db", "token": {

"symbol": "ETH",

"address": "0x153F602ad18BBD1546b674cFDBe05a8ba72A1d57"

},

"type": "redeem",

"amount": "531.208420",

"shares": "2028.123080",

"timestamp": 1524737174690

}

],

"audits": [

{

"auditor": "0x9b8C165672b41725817a606c18C117C5a171D96b",

"dataHash": "1Nf2mjaHRhZK3qf9LrSveKimqVx3vUau5q", "timespanStart": 1524729021140,

"timespanEnd": 1524739020045

},

{

"auditor": "0x9b8C165672b41725817a606c18C117C5a171D96b",

"dataHash": "1MNDVGk51Wyty9YqjaZb99PDAPj4R2wEgx", "timespanStart": 1524729086440,

"timespanEnd": 1524739020999

}

]

}

# Auditing Contract

## Considerations

There are multiple ways to store audits, each with their own strengths and weaknesses. Storage also has an effect on crucial function calls like *isComplete*. In this chapter, we compare the theoretical complexity as well as the estimated gas costs of four variants. We also balance the pros and cons considering standard auditing processes and flexibility of the implementation on timespans.

There are two main considerations:

The cost of storing an audit (by *add*) must be as low as possible

We want to incentivize auditors to properly audit funds regarding the timeline. This means that they should be encouraged to never produce gaps in audited timespans of a fund and audit in a linear fashion. When they audit steadily, they should expect steady gas costs when calling the *add* function. When they have to close gaps in the audited timespan, they should not be punished too hard by gas costs.

Validating *audit completeness* of a fund over a timespan (by *isComplete*) must be as efficient as possible

Audit completeness of a fund over a timespan will not only be verified manually by investors. Melon risk management modules will also rely on completeness verification in the future. A risk module may prevent a fund manager to trade when the fund has not been audited recently (see [Ex Ante](http://www.docs.melonport.com/chapters/risk_engineering.html#ex-ante---engineering)). As these risk modules are calls from smart contracts, the *isComplete* function must be as efficient as possible in regards to gas cost and method invocation time.

### Auditing Contract Variants

We came up with the following variants in regards to the contract data structures:

Variant 1 - Array with shifting indices

Variant 2 - Helper array of audited timespans Variant 3 - Linked list

Variant 4 - Fixed time periods

### Scenarios regarding complexity

TODO graphs for cases!

### add

Worst case scenario: Audit is added to start Average case scenario: Audit closes gap in middle Best case scenario:

Audit is added to end

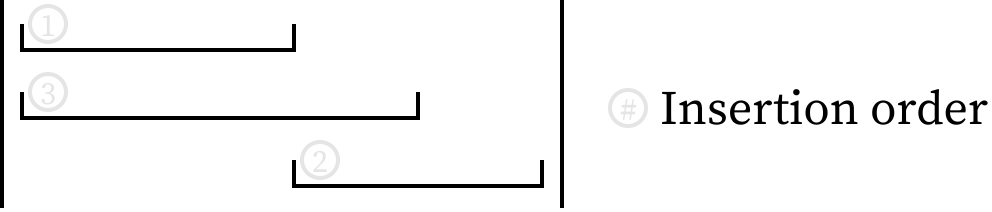
### isComplete

TODO: maybe do this directly in variants because contracts have different worst cases... Worst case scenario: Average case scenario: Best case scenario:

### Practical and theoretical cases

Theoretical complexity does not suffice for our scope, because gas cost on average and best case performance will vary a lot, and these normally have a complexity of *O*(1) in theory. For this reason, we extend Big-O notation with more precise values like Ω(7) for 7 rather gas-intensive operations. We call this *practical complexity*.

### Variant 1 - Array with shifting indices

****

There is one array per fund, storing all the audits. The audits are sorted by *timespanEnd*. Strengths:

Array is always sorted Weaknesses:

When an audit with a very early timespanEnd is inserted, the gas cost rises rapidly because a lot of indexes have to be shifted.

Indexes to audits can change

### add complexity

##### Worst case

Theoretical: *O*(*n*) - Practical: *O*(2*n* + 1)

Look for insertion position: O(n) Shift indices: O(n)

Insert new timespan: O(1)

##### Average case

Theoretical: Θ(*n*) - Practical: Θ(*n* + 1)

Look for insertion position: O(n/2) Shift indices: O(n/2)

Insert new timespan: O(1)

##### Best case

Theoretical: Ω(1) - Practical: Ω(2)

Compare last value and break out of loop: O(1) Add audit to end of array: O(1)

### isComplete complexity

##### Worst case

Theoretical: *O*(*n*) - Practical: *O*(*n*)

Compare all values in array: O(n)

##### Average case

Theoretical: Θ(*n*) - Practical: Θ(*n*/2)

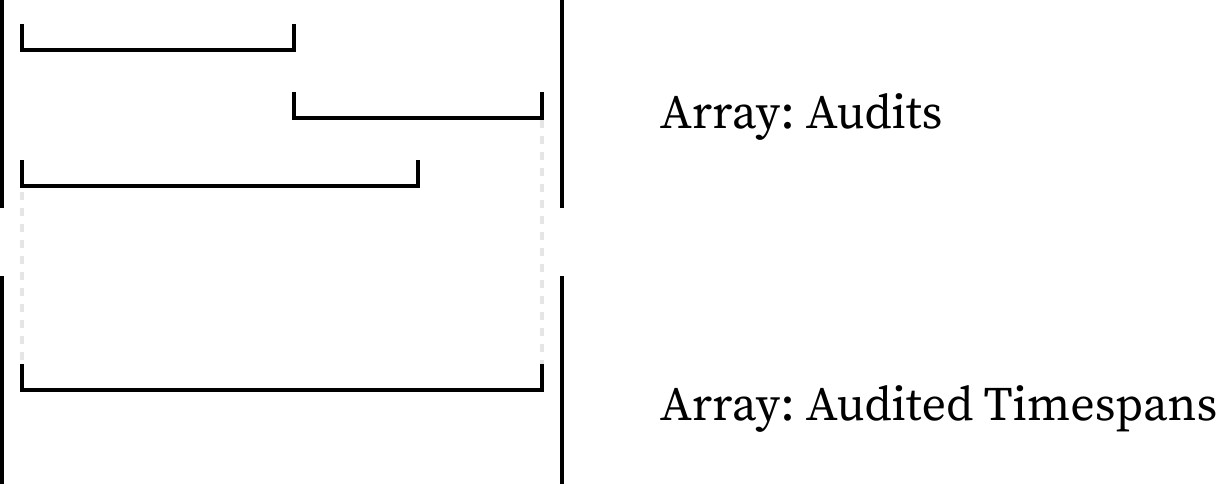
Compare half of the values in array: O(n/2)

##### Best case

Theoretical: Ω(1) - Practical: Ω(1)

First audit covers given timespan: O(1)

### Variant 2 - Helper array of audited timespans

****

An array *timespanAudited* with (a struct of?) timespanStart and timespanEnd. Best case is that only one timespan is in it: From when to when the fund is audited. But if there are gaps, the array holds more values.

When we use the *isComplete* function, we just have to check this array, so we don't have to iterate over a lot of audits and their timespans.

Strengths:

Cheap isComplete call (best case: only testing 2 values) Indexes always stay the same

Standard add is cheap (case: fund is well audited without gaps) Weaknesses:

Audit array is not sorted by timespans.

### add complexity

##### Worst case

Theoretical: *O*(*n*) - Practical: *O*(3*n* + 1)

Shift and insert new timespan: O(2n) Merge all previous timespans: O(n) Add audit to end of array: O(1)

##### Average case

Theoretical: Θ(1) - Practical: Θ(9)

Look for insertion position with two timespans present: O(2) Shift index of second timespan: O(1)

Insert new timespan: O(1)

Merge three timespans to one (delete two, change one): O(3) Add audit to end of array: O(1)

##### Best case

Theoretical: Ω(1) - Practical: Ω(5)

Look for insertion position one timespan present: O(1) Add new timespan to end of timespan array O(1) Merge two timespans (delete one, change one): O(2)

Add audit to end of array: O(1)

### isComplete complexity Worst case

Theoretical: *O*(*n*) - Practical: *O*(*n*)

*n* audits produced *n* gaps, compare all: O(n)

##### Average case

Theoretical: Θ(1) - Practical: Θ(2)

One gap, check two timespans: O(2)

##### Best case

Theoretical: Ω(1) - Practical: Ω(1)

No gaps, check one timespan: O(1)

### Variant 3 - Linked List

[Blog post: Linked Lists in Solidity](https://medium.com/coinmonks/linked-lists-in-solidity-cfd967af389b) Strengths:

Low gas cost for special case: *insert audit in between existing audits*

Weaknesses:

Iterating through the linked list is expensive

We cannot access audits by index, only if we would create the indexes on the fly

*On the fly* indexes would also change after an insertion

As we did not implement a linked list variant in [solidity](#_bookmark81), we will argue about the theoretical complexity of the data structure [compared to the dynamic array variants](https://en.wikipedia.org/wiki/Linked_list#Linked_lists_vs._dynamic_arrays).

The linked list variant could benefit from a separate timespan array like in variant 2. This would make the audit completeness check much more efficient with comparably small extra effort.

### add complexity

##### Worst case

Theoretical: *O*(*n*)

##### Average case

Theoretical: Θ(*n*)

##### Best case

Theoretical: Ω(1)

Assumption: *head* of list is known

### isComplete complexity

##### Worst case

Theoretical: *O*(*n*)

##### Average case

Theoretical: Θ(*n*)

##### Best case

Theoretical: Ω(1)

### Variant 4 - Fixed time periods

We could only allow audits to be performed on whole months or other fixed timespans. We could store them very easily in a map.

Mapping could be done with *year => Audit[12]* or similar. Strengths:

No timespans, so no possibility of off-by-one errors

Gaps are very easily detectable Indexing is easy

Weaknesses:

No flexibility for audit timespans

Indexes do not align with *audits done* when there is a gap. We could get an audit for indices 0 and 2, but not for index 1.

isComplete is more expensive than with variant 3

The fixed time periods variant could also benefit from a separate timespan array like in variant 2. This would make the audit completeness check much more efficient, but would not change the fact that time periods are not flexible.

As we did not implement a variant with fixed time periods in [solidity](#_bookmark81), we will argue about the theoretical complexity of the [hash table data structure](https://en.wikipedia.org/wiki/Hash_table).

### add complexity

##### Worst case

Theoretical: *O*(*n*)

##### Average case

Theoretical: Θ(1)

##### Best case

Theoretical: Ω(1)

### isComplete complexity

##### Worst case

Theoretical: *O*(*n*)

##### Average case

Theoretical: Θ(1)

##### Best case

Theoretical: Ω(1)

### Considering reality Variant 1

The likeliness to compare all values on isComplete is very high (example: check [inception](#_bookmark73) to now), which means that the worst case would happen very often.

### Variant 2

We expect the auditors to add audits in a linear fashion in regard to timespans without introducing a lot of gaps.

*Variant 2* would perform very good on completeness verification.

### Variant 3

This variant would introduce a new data structure. With this, the contract would be a lot more complex without providing crucial benefits.

### Variant 4

Flexibility would be very low with this solution. Auditors would be restricted to a period. Also, multiple audits on the same period would not be possible this way.

### Decision

Considering both theoretical and practical aspects, *Variant 2 - Helper array of audited timespans* is the most flexible, cost efficient and practical implementation.

The benefits of this overweigh the slightly higher complexity on adding an audit.

As *Variant 1 - Array with shifting indices* is very similar to *Variant 2*, we implemented both versions. Let us look at the implementation as well as resulting gas costs and method invocation complexity through tests.

## Implementation

TODO

## Variant specific tests

### insertAudit Gas cost

This is the default gas value that a block currently can hold: **Block gas limit: 8'000'029** [(Source)](https://ethstats.net/) TODO IMG (Screenshot from 2018-06-15)

### Variant 1

Gas cost for adding one audit when **one** audit is present:

|  |  |
| --- | --- |
| **Case** | **Gas cost** |
| Gas cost "add to array end" (no index shift) | 201501 |
| Gas cost "add to array start" (one index shift) | 256201 |
| Extra gas cost for index shift (one shift - no shift) | 54700 |

Gas cost for adding one audit when **ten** audits are present:

|  |  |
| --- | --- |
| **Case** | **Gas cost** |
| Gas cost "add to array end" (no index shift) | 201573 |
| Gas cost "add to array start" (ten index shifts) | 765043 |
| Extra gas cost for index shift (ten shifts - no shift / 10) | 56347 |

Gas cost for adding one audit when **100** audits are present:

|  |  |
| --- | --- |
| **Case** | **Gas cost** |
| Gas cost "add to array end" (no index shift) | 201573 |
| Gas cost "add to array start" (hundred index shifts) | 5853463 |
| Extra gas cost for index shift (hundred shifts - no shift / 100) | 56519 |

Side note: "add to array start" implies that **X** values have to be shifted.

Adding an audit to the start of the array is a potential problem with *Variant 1* considering the block gas limit, even if the auditors have audited the fund in a linear fashion beforehand (i.e. they did not produce gaps).

### Variant 2

Gas cost for adding one audit when **one** audit is present:

|  |  |
| --- | --- |
| **Case** | **Gas cost** |
| Gas cost "add latest timespan" | 369429 |
| Gas cost "add earliest timespan" | 394581 |
| Extra gas cost | 25152 |

Gas cost for adding one audit when **ten** audits are present:

|  |  |
| --- | --- |
| **Case** | **Gas cost** |
| Gas cost "add latest timespan" | 369429 |
| Gas cost "add earliest timespan" | 394581 |
| Extra gas cost | 25152 |

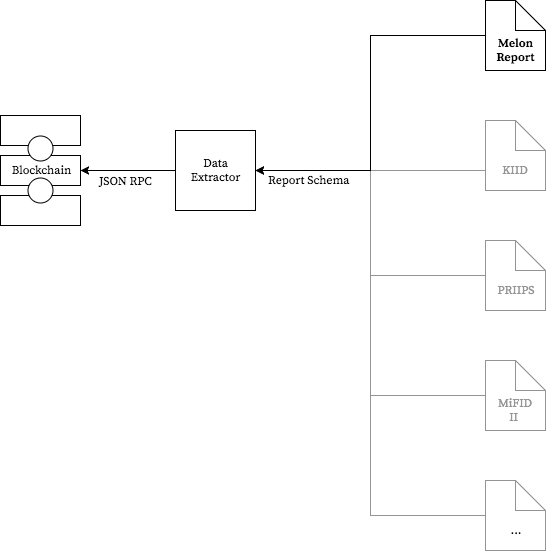
Gas cost for adding one audit when **100** audits are present:

|  |  |
| --- | --- |
| **Case** | **Gas cost** |
| Gas cost "add latest timespan" | 369429 |
| Gas cost "add earliest timespan" | 394581 |
| Extra gas cost | 25152 |

With *Variant 2*, adding an audit as the earliest in regards to the timespan is not a gas limit issue.

### isComplete Gas cost

**Deterministic Data Extraction**

****

**Explanation**

A function that extracts all necessary data for a fund report from the blockchain in one go:

extractData(fundAddress, timespanStart, timespanEnd): FundReportJSON

The function is deterministic: For the same arguments it returns always the same result. The result is a valid JSON according to the Fund Report Data Schema.

This JSON is the foundation for our Standard Report Web Interface but could also be used for different types of reports: [KIID](#_bookmark75), [MiFID II](#_bookmark76), etc.

## Calculations

Assumption: A fund setup never changes. All settings on fund creation are persistent. This might change in the future (see Future Work).

DailyPrice (see Decisions) Benchmark (Reference "Fund")

Total number of shares (*s*) can be retrieved with the following formula:

*s* = ∑ *qi*

*i*∈*I*

Where *I* is the set of all investments and *qi* the quantity of shares per investment.

[AUM](#_bookmark64) (*aum*) can be retrieved with the following formula:

*aum* = ∑ *ha* ∗ *pa*

*a*∈*A*

Where:

*A* is the set of all assets a fund is invested in

*ha* the amount of holdings the fund has of asset *a pa* the price of asset *a*

Shareprice (*sp*):

*sp* = *aum*

*s*

## Standard Report Web Interface

The final step is to visualize the extracted data in a nice looking report

Takes a standard report data JSON as input and visualizes it

This decouples the web interface from the rest of the system: It could just be mock-data.

The web interface follows the mockup as close as possible

Implementation of the Mockup as a web interface

URL Scheme: /report/:fundAddress/:timeSpanStart/:timeSpanEnd

# Conclusion

**Findings**

(show whole reporting and auditing cycle with pictures)

(show that requirements are met and stakeholder needs were considered)

# Future work

## Legal Entity and Strategy Data Extraction

use *Volume Weighted Average Price*

consider "changing whitelists" use *Fund alpha*

use other metrics like *Max Drawdown*

Green ticks for KYC investors

# Reflection

**Technical Quickstart**

**Project management**

## Project Goals

### Interchangeable Fund Data Format

**Auditing Smart Contract Interface Specification**

**Auditing Smart Contract Interface Reference Implementation Interchangeable Fund Data Extraction and Consolidation Service Fund Report Web Interface With Auditing Functionality**

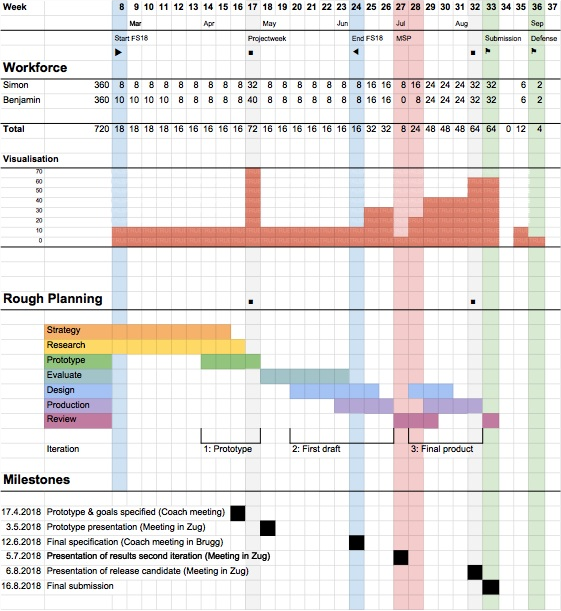
**Summary of Meetings**

27.3.2018 - 3pm - In Brugg with coaches: 5.2B31 17.4.2018 - 3pm - In Brugg with coaches: 5.2B31

3.5.2018 - 2pm - In Zug with all stakeholders except Sarah Hauser 22.5.2018 - 4pm - In Brugg with Markus Knecht: 5.2B31 12.6.2018 - 3pm - In Brugg with all coaches: 5.2B31

5.7.2018 - 2pm - In Zug with all stakeholders 17.7.2018 - 3pm - In Brugg with coaches: 5.2B31 6.8.2018 - 2pm - In Zug with all stakeholders 13.8.2018 - 3pm - In Brugg with coaches: 5.2B34

# Planning



## Strategy

In a first step the foundation of the project is laid. Vision, hypothesis and boundaries are defined and agreed upon with the stakeholders and a rough planning is created. Furthermore repositories are created, docs initialized, etc.

**Deadline**: 13.4.2018

##### Expected results:

Project vision Hypothesis

Project boundaries Detail planning

Basic setup of environment

Documentation outline

## Research

The next step is to deep dive into the problem domain. We read and summarize the important articles about the topic and learn the underlying technologies. Also part of research are interviews.

**Deadline**: 20.4.2018

##### Expected results:

Overview & summary of material: Articles, law, templates, ...

Knowledge of underlying technologies (Blockchain, [Solidity](#_bookmark81), React, Redux, digital signing, etc.) Transcripts of interviews

## Prototype

The collected knowledge from the research phase is now transformed into a first prototype which can be challenged by the stakeholders and test users.

**Deadline**: 27.4.2018

##### Expected results:

Prototypes Wireframes

## Evaluate

Testing & discussing the prototype with the stakeholders and test users give us valuable insights for the further development.

**Deadline**: 22.6.2018

##### Expected results:

Evaluation reports User testing reports

## Design

Already during the evaluation phase we start the design phase to have an iterative process: The prototype and wireframes are adjusted from the feedback but also the work on the final mockups and software architecture is started.

**Deadline 1**: 22.6.2018 **Deadline 2**: 27.7.2018

##### Expected results:

Mockups

Software Architecture Specifications

## Production

Iterative development and finishing a release candidate. This will also take place in combination with design and review phase. Basically in the following loop: Design -> Production -> Review -> Design -> ...

**Deadline 1**: 29.6.2018 **Deadline 2**: 10.8.2018

##### Expected results:

Final product first and second version

## Review

Collect feedback of the release candidate, finish documentation and submission of the thesis.

**Deadline**: 17.8.2018

##### Expected results:

Review reports Final thesis report

# Milestones

## 17.4.2018 - Prototype & goals specified

Coach meeting in Brugg

Specified what's in the prototype and what not General project goals specified with Melonport

Project plan with more detailed & project oriented planning Documentation outline fixed

## 3.5.2018 - Prototype presentation (Meeting in Zug)

Presentation of the finished prototype in Zug All stakeholders invited

## 5.7.2018 - Presentation of results second iteration (Meeting in Zug)

Presentation of the results of second iteration in Zug All stakeholders invited

First draft of documentation submitted to coaches Finalize specification about final product

Requirements fixed

All documents available All data available

All prototypes ready for extension

## 6.8.2018 - Presentation of release candidate (Meeting in Zug)

Presentation of the release candidate in Zug

Collection of last feedback and adjustments for final release All stakeholders invited

## 16.8.2018 - Final submission

Official submission of thesis as bound paper Final version deployed

## Iterations

### Iteration 1: Prototype

The first iteration is a functional prototype. The goal is to have something clickable as soon as possible. For this phase, it is not yet important to have all fields and data. But something that can be shown to fund managers to collect first valuable feedback.

**Deadline**: 27.4.2018

### Iteration 2: First draft

The second iteration aims already at the final goal & specification knowing that it is not possible to cover all topics yet. Still, it should be as functional as possbile to gather more detailed user feedback already.

**Deadline**: 29.6.2018

### Iteration 3: Final product

Finally, all feedback is collected and the final product can emerge from the first draft.

**Deadline**: 10.8.2018

# Journal

We send a short status update every week to our stakeholders. They are also stored here for reference.

## Calendar Week 8

We started working on the thesis and made a first broad overview over the topics:

Benjamin started to research into blockchain development particularly [Solidity](#_bookmark81).

Simon setup the repository and got in touch with possible project supporters from PwC and read a bit into the domain of legal reporting of collective investment schemes.

## Calendar Week 9

In the second week we already deep dived into the specific domains:

Benjamin set up the [Solidity](#_bookmark81) development environment according to the Melon setup with dapp.tools, parity dev chain but also looked into truffle suite.

Simon researched [MiFID II](#_bookmark76) and [PRIIP](#_bookmark78) and started the [glossary](https://github.com/melonproject/reporting-thesis/blob/master/docs/GLOSSARY.md) for these confusing abbreviations. Furthermore, he finished the strategy part.

## Calendar Week 10

This week we had the official kick-off meeting with all stakeholders. See minutes in [Appendix](https://schmidsi.gitbooks.io/melon-reporting/content/Appendix.html).

Simon updated the project plan according to the feedback and discussed the [KIID](#_bookmark75) template provided by PwC with Mona

Benjamin further read into [Solidity](#_bookmark81), especially data structures like strings and byte64.

## Calendar Week 12

A lot of progress is already visible in the different areas:

Benjamin created a very rough [working first draft of the auditing contract](https://github.com/melonproject/reporting-thesis/blob/f3da136beab45b6683dda5ec4001f5415c877870/packages/contracts/Auditing.sol)

[Simon visualized the logical data structure and started to derive different report requirements according to these structures.](https://melonproject.github.io/reporting-thesis/02-research/)

Furthermore, we had our first coach meeting.

## Calendar Week 15

Benjamin started with the auditing interface proposal and it's implementation. Furthermore, he created a rudimentary web interface to this auditing contract.

Simon refined the documentation outline of the report, created the prototype specification proposal and started to walk through the [FINMA](#_bookmark72) [KIID](#_bookmark75) document. Furthermore he met with Jenna Zenk and John Orthwein to sync risk management with reporting.

The research takes longer than expected, therefore we jumped forward for the prototype and will implement directly a standard report mockup draft without proper written research foundation. Simon already knows in his head how such a standard report needs to look like. More validating research can also be done later. It is important to have something to show now rather than thorough research.

## Calendar Week 17: Project Week

*Review project week and meeting invitation*

Hello everybody

This week we worked intensely on the thesis and achieved our goals. We will discuss them with you at our next big meeting:

Thursday, 3. Mai 14:00 at our offices at Park Hotel, Industriestrasse 14, 6300 Zug. Until then, here is a quick

##### Summary

*Use cases*

We created more detailed use cases which are part of our requirements engineering that should act as the official goals for our thesis. I pasted them at the end of this mail.

*Auditing Contract Standard Proposal*

We submitted a [Melon Improvement Proposal](https://github.com/melonproject/MIP/issues/1) to the MIP repository so that all other developers and interested parties see what we are up to and can join the conversation. I pasted the text at the end of this mail.

*Report Mockup*

We created a full report mockup that shows in a tangible way how Melon Fund reporting will look like in the future. I attached a PDF export to this mail.

*Fund Report Data Schema*

Also, we wrote a JSON Schema and a reference file how the underlying data of future fund reports look like. This is also pasted at the end of this mail.

*Reorganise documentation*

Furthermore, we reorganized the documentation as suggested by our coaches.

##### Agenda for the meeting

Present & discuss results from prototype iteration Decide project goals (use cases)

Questions & discussion

##### Questions

We already have the following questions:

Volatility for reports: from [inception](#_bookmark73) to timespanEnd or from timespanStart to timespanEnd? Risk Mgmt: is data immutable?

priceHistory: is daily interval good?

priceHistory: which value? mean or median or snapshot? Address to real name lookups on-chain?

## Calendar Week 19

Hello everybody

### Minutes

Last week we met in Zug for the presentation and a fruitful discussion of our prototype work. You find the minutes [online](https://melon-reporting.now.sh/02-appendices/E-minutes/) but also copy/pasted to this mail for your reading convenience.

### Work

We updated the [project planning](https://github.com/melonproject/reporting-thesis/projects/1) according to the feedback from this meeting

We added [decisions](https://melon-reporting.now.sh/02-appendices/B-projectmanagement/06-Decisions.html) derived from this meeting. Also pasted to this mail. The main decision is that we will focus on on-chain data but document the interface how in the future off-chain data (e.g. from IPFS) could also be used.

Benjamin deployed the auditing prototype smart contracts to kovan and the simple front-end to now: <https://auditing-frontend-hccgfdqbgw.now.sh/>. Please make sure that you have MetaMask installed if accessing this link otherwise an ugly error will appear.

### Small planning shift

We realized that there is too little time and resources left until the end of the second iteration on 5th July. But we already have two counter measures:

We promised to also hand in a first draft of the report to the coaches by 5th July. We decided that the main focus for this iteration should be the product and not the report. So we'll focus on building the product and will work on the report later on: **The new deadline for the first draft of the report is: 11th July**, so that the coaches have enough time to read it until the coach meeting on 17th July.

Simon will try to synchronize the general Melon-planning with the thesis planning better in order to move some days from the final sprint in August to June.

### Next steps

**All**: Find more users for explorative user testings / interviews. If you know someone who is interested in our work and could have good feedback, please send me the contacts.

Enhance mockup with all the feedback. The mockup will have much more features on it as we will actually implement. This is by intention: The mockup should also guide and inspire future work.

Start with the real product:

Deterministic data extraction module with GraphQL endpoint Report rendering/visualization

**Next milestone: 12th June: Final specification** presented to the coaches in Brugg.

# Risks

|  |  |  |
| --- | --- | --- |
| **Name** | **Counter measures** | **Risk (\*)** |
| Distractions from classes | Clear timeboxing. Clear planning. Buffers. | 2 \* 1  = 2 |
| Distractions from job | Clear timeboxing. Clear planning. Clear and upfront communication of availabilities. Support from management. | 1 \* 1  = 1 |
| Project team out of sync | Weekly team work time slots. Open communication. Weekly reports. | 2 \* 2  = 4 |
| Dependency on systems out of project scope | Apply subsystem decomposition and isolation techniques from the beginning. | 3 \* 2  = 6 |
| Losing focus / distraction by details | Weekly reports to coaches and project sponsors to gather feedback. | 3 \* 2  = 4 |
|  |  |  |

(\*) Probability of occurrence (1-3) \* severity (1-3) = risk

**Stakeholders Melonport (Customer) University**

## Coaches Project team

**Decisions**

**2018-04-26**

### Use Cases

For the Use Case diagram, we decided to exclude the function "Comment timespan". A fund manager would have been able to save an off-chain comment about a timespan which would later on show in a report. However, it is not possible to include something like this in the report, because adding such a comment later on or changing the comment would change the hash of the document and would therefore invalidate past audits. If a fund manager wants to communicate with their investors through these comments, there are other ways to do this, e.g. with blog posts or through Melon Mail.

## 2018-05-08

Discussing the prototype meeting, we decided the following

As *daily price* we use the mean over all prices per asset that the pricefeed delivered from 00:00:00 UTC to 23:59:59 UTC.

Different auditors will be restricted to different audit classes in the contract.

*Style*, *Substyles*, *Strategy*, *Substrategies* will be future work (will not be part of the JSON)

*Fund Alpha* will be future work.

Benchmark is *Weighed Average Share Price*

Only work with on-chain data but we specify the possibility to work with off-chain data too. But this implementation and detail specification is moved to future work.

# Iterations

**Prototype**

**Goals**

The purpose of the prototype is to build something early that can already be shown to different stakeholders to collect feedback. This can be in the form of: Click dummies, wireframes, mockups, specifications or actual functional prototypes.

## Specification

We suggested the following specifications to the customer and coaches and they approved them (TODO).

### Must

These are the minimum tasks that need to be done after the project week.

### Melon Auditing Contract Standard Proposal

Introduce new functionality to the community is through a [Solidity](#_bookmark81) interface specification proposal. In Ethereum they are called [ERC](#_bookmark68). We create an interface specification proposal for the auditing contract so that the core Melon smart contract developers can see the direction we are going and give early feedback.

A good example is the [ERC-223 Token Standard Proposal](https://github.com/ethereum/EIPs/issues/223) We will submit this as a [MIP (Melon Improvement Proposal)](https://github.com/melonproject/MIP/issues)

### Standard Report Mockup Draft

To have something to show potential users of the system we create a fictional report that is inspired by the legacy reports that we researched in chapter "Research". This will be in the form of a fictional mockup draft: The deliverable is a static image export from [Sketch](#_bookmark80):

All applicable information from [KIID](#_bookmark75) but also timely reports. E.g. Strategy, [SRRI](#_bookmark82), past performance, benchmark, ... All applicable information from the blockchain. E.g. Holdings, recent trades, fund name, risk engineering settings,

...

Draft means: Not a pixel perfect design, but all information already visible or indicated. Fictional: We do not work with real data but we invent data that could be real.

### Interchangeable Fund Data Format Draft

Create a general interchangeable fund data format that is the basis for all the different reports and also the data that is actually hashed and signed by the auditing contract.

A annotated JSON Schema and/or GraphQL Schema An example JSON

### Bonus

The "must" tasks are estimated pessimistic so there should be time left for some of the following tasks:

### Deploy Auditing Contract Draft To Kovan

Deploy a first draft of the auditing contract to Kovan and link it to the current Melon smart contracts.

Auditing smart contracts coded and ready to deploy Deployed to Kovan

### Auditing Web Interface Draft

Write a simple web interface to interact with the auditing smart contracts.

### Standard Report Web Interface Draft

Start with the web interface that can transform the interchangeable fund data into a standard report.

# Implementation

## First draft of auditing contract

In a rough first draft for the auditing part of the project, we had the idea of storing the datahash of the audit along with a signature. To proof an audit as legit, the auditor had to create the signature with web3.eth.sign [Citation not found]

v

and split it up into the values , , . The auditor then called the function audit of the auditing smart contract

r

s

with the *fund address*, the *datahash* and the values *r*, *s* and *v*. On the [Solidity](#_bookmark81) contract, it was a requirement that the provided signature is valid before storing the audit data on the ledger. With the function checkSignature , we recovered the datahash and signature values with the elliptic curve recovery function ecrecover [4] to the auditor address. This address has to be equal to the sender address.

After receiving feedback from Markus Knecht about this approach, we discussed the necessity of signing the audits "by hand". Basically, a transaction on the blockchain is by itself a signature. The only reasonable use case of an additional signature is when additional participants have to identify themselves. An example for this is the [0x protocol](https://0xproject.com/) where on-chain data and off-chain data is mixed. This is not the case for auditing (there is only one auditor), so we dropped this functionality for the prototype.

## Prototype design Auditing contract

To receive feedback from the Melonport staff about our prototype implementation of the auditing contract, we set up a [MIP (Melon Improvement Proposal)](https://github.com/melonproject/MIP/issues/1)

## Test frontend for the auditing contract Melon Fund Report Data Schema

We replaced the first name of the report data schema from *Interchangeable Fund Data Format* to *Melon Fund Report Data Schema* (M) because it describes its use better.

### Ethereum Address

Ethereum addresses have the following attributes:

starting with 0x

followed by 40 hex characters

This considered, we defined this regular expression to validate Ethereum addresses:

^0x(\\d|[A-F]|[a-f]){40}$

*Note*: The uppercase chars are checksums described in [EIP-55](https://github.com/ethereum/EIPs/blob/master/EIPS/eip-55.md)

### ERC20 token symbols

For ERC20 token symbol standards, we found this discussion: [Link](https://ethereum.stackexchange.com/questions/25619/is-there-length-limits-on-token-symbols)

So there are currently only symbols on the chain with the following attributes:

type: string length: 1 to 9

all uppercase letters

This considered, we defined this regular expression to validate ERC20 token symbols:

^([A-Z]){1,9}$

### Legal Entity and Strategy

Legal Entity and Strategy are optional, because this data is not (yet) saved on the blockchain. It will be part of the "future work" chapter.

### Big Numbers

In Ethereum, big numbers are often used. For melon, we see this in places like "token quantity". These numbers will have the following attributes:

type: string

numbers followed by a dot (.) followed by numbers

This considered, we defined this regular expression to validate big numbers:

^\d\*.\d\*$

### Market Cap Range

The "max" value of marketCapRange is not required. When none is supplied, the property is undefined, which means "max" is (positive) infinity.

### Fund Report Data Example

This was a rough first draft of an *FRDS* instance:

{

"name": "Example Fund", "inception": "yyyy-mm-dd hh:mm:ss",

"description": "This fund is high risk", "manager": "0xbad...a55",

"nav": 1000,

"quoteSymbol": "MLN", "gav": 1100,

"timestamp": "yyyy-mm-dd hh:mm:ss", "holdings": [

{

"symbol": "ETH", "amount": 1000

}

],

"trades": [

{

"buySymbol": "ETH",

"sellSymbol": "MLN", "buyAmount": 100,

"sellAmount": 50,

"timestamp": "yyyy-mm-dd hh:mm:ss",

"market": "0xdead...beef"

}

],

"audits": [

{

"timestamp": "yyyy-mm-dd hh:mm:ss",

"auditor": "0xdead...beef1",

"dataHash": "QmXZcdco6wZEA2paGeUnoshSB4HJiSTDxagqXerDGop6or", "signature": "0x23rasdfasdlfjhasldkfhas"

}

]

}

This is the example data after the prototype schema definition, following the rules of the schema v0.1:

// copy paste example here

# Internal

**Stakeholder Feedback**

**Explorative User Testings**

**Conclusion**

**First draft**

**Goals**

The subsystems in System Overview are the goals of the second iteration that we propose. The general goal is a fully functional product. I.e. 80% according to the Pareto principle. With the outcome of this iteration we should be able to do real user testings and gather feedback from a broad audience. So that we can implement this feedback in the last iteration.

Besides the subsystems from *System Overview*, here follow the rest of the goals.

## Deterministic Data Extractor

As described in chapter Deterministic Data Extraction

### Boundaries

We aim to work with **@melonproject/smart-contracts@~0.8.0**, the version for the Paros Olympiad if possible. If not, we fall back to whatever is deployed with useful data on Kovan or Main Chain.

## Standard Report Web Interface

As described in chapter Standard Report Web Interface

### Boundaries

Recommended: Add positions (open/close & P&L per position) Optional: ENS Name lookups

Optional: KYC indicators

Optional: Printing functionality. It is not sure that printing will work without problems. Optional: Sorting and other interactions

Without risk management and other data/fields that are not implemented in the smart contracts yet. I.e. Strategy, category, ...

## Auditing Web App

Functionality on top of the Standard Report Web Interface to add an audit to the viewed data. We sign the contract data with Metamask.

## Finalizing

We will finalize the work from the prototype iteration according to the received feedback:

Report Schema Standard Report Mockup

Auditing Contract Standard

Auditing Contract Reference Implementation

## Yet unknown

Feedback from industry stakeholders (e.g. PwC, Deloitte, SFAMA and others) will be assessed when available and either directly implemented if possible or documented in chapter "Future work".

## Optional

Depending on the outcome of the above tasks, we might do one of the following if time permits and it is meaningful for the stakeholders.

### Fund Database

If the Report Data Extractor either runs too slowly or is too complicated to implement without a database, we can create as a fallback but also as a performance and user experience improvement a database that holds all data about funds for reports.

Complete database schema for funds according to the [ERM](#_bookmark69).

Methods (queries) to get the same data as in Report Data Extractor: queryReportData(fundAddress, timespanStart,

timespanEnd): FundReportJSON

#### 

extractData(fundAddress, timespanStart, timespanEnd) === queryReportData(fundAddress, timespanStart,

timespanEnd)

Mutations to add data to the database. I.e.

|  |  |  |
| --- | --- | --- |
| addFund |  | |
| updatePrice | |  |
| addParticipation | | |
| addTrade |  | |

More if necessary

### Blockchain Sync Service

A service that watches the blockchain for relevant events and updates the Fund Database accordingly Read the current state of the Melon smart contracts on startup

Sync the changes

Optional: Possibility to shut service down and restart it and it can sync the missed blocks.

### Report database

An easier way to enhance the user experience for the use case "View Report" is to just create a database where auditors store the extracted report data once audited. The other actors can then easily access these reports and check their validity simply by executing the function Audit.exists(fundAddress, dataHash) .

# Implementation

**Validative User Testing**

**Auditing Contract**

**Deployment cost**: less than 791800 gas

## Constructor

Params:

1. **\_approvedAuditors** *of type address[]*

## Events

### Added(address,uint256)

**Execution cost**: No bound available Params:

1. **\_fundAddress** *of type address*
2. **\_index** *of type uint256*

## Methods

### getAuditedTimespansLength(address)

**Execution cost**: less than 738 gas

**Attributes**: constant Params:

1. **\_fundAddress** *of type address*

Returns:

1. **length** *of type uint256*

### auditedTimespansPerFund(address,uint256)

**Execution cost**: less than 1092 gas

**Attributes**: constant Params:

1. **param\_0** *of type address*
2. **param\_1** *of type uint256*

Returns:

1. **start** *of type uint256*
2. **end** *of type uint256*

### approvedAuditors(uint256)

**Execution cost**: less than 1057 gas

**Attributes**: constant Params:

1. **param\_0** *of type uint256*

Returns:

1. **output\_0** *of type address*

### add(address,bytes32,uint256,uint256,uint256)

>

Creates a new audit on a fund specified with \_fundAddress , the hashed data in \_dataHash1 and \_dataHash2

and the timespan timestamps in \_timespanStart and \_timespanEnd .

**Execution cost**: No bound available Params:

1. **\_fundAddress** *of type address*
2. **\_dataHash** *of type bytes32*
3. **\_timespanStart** *of type uint256*
4. **\_timespanEnd** *of type uint256*
5. **\_opinion** *of type uint256*

### exists(address,address,bytes32)

>

Validates that the provided data is mapped to an existing audit

**Execution cost**: No bound available

**Attributes**: constant Params:

1. **\_fundAddress** *of type address*
2. **\_auditor** *of type address*
3. **\_dataHash** *of type bytes32*

Returns:

1. **auditExists** *of type bool*

### fundAudits(address,uint256)

**Execution cost**: less than 2192 gas

**Attributes**: constant Params:

1. **param\_0** *of type address*
2. **param\_1** *of type uint256*

Returns:

1. **auditor** *of type address*
2. **dataHash** *of type bytes32*
3. **timespanStart** *of type uint256*
4. **timespanEnd** *of type uint256*
5. **opinion** *of type uint8*

### getAuditedTimespanEnd(address,uint256)

**Execution cost**: less than 993 gas

**Attributes**: constant Params:

1. **\_fundAddress** *of type address*
2. **\_index** *of type uint256*

Returns:

1. **end** *of type uint256*

### getAuditedTimespanStart(address,uint256)

**Execution cost**: less than 1015 gas

**Attributes**: constant Params:

1. **\_fundAddress** *of type address*
2. **\_index** *of type uint256*

Returns:

1. **start** *of type uint256*

### getByIndex(address,uint256)

>

Returns the requested audit data **Execution cost**: less than 2967 gas **Attributes**: constant

Params:

1. **\_fundAddress** *of type address*
2. **\_index** *of type uint256*

Returns:

1. **auditor** *of type address*
2. **dataHash** *of type bytes32*
3. **timespanStart** *of type uint256*
4. **timespanEnd** *of type uint256*
5. **opinion** *of type uint256*

### getLength(address)

>

Returns the length of the audit array of a specific fund

**Execution cost**: less than 851 gas

**Attributes**: constant Params:

1. **\_fundAddress** *of type address*

Returns:

1. **index** *of type uint256*

### isApprovedAuditor(address)

**Execution cost**: No bound available

**Attributes**: constant Params:

1. **\_auditor** *of type address*

Returns:

1. **auditorIsApproved** *of type bool*

### isComplete(address,uint256,uint256)

>

Returns true if a fund is completely audited over a specific timespan.

**Execution cost**: No bound available

**Attributes**: constant Params:

1. **\_fundAddress** *of type address*
2. **\_timespanStart** *of type uint256*
3. **\_timespanEnd** *of type uint256*

Returns:

1. **complete** *of type bool*

# Final product

**Changes to first draft**

**Research**

Approach:

1. Research & classify available data
2. Research report types ([KIID](#_bookmark75), PRIIPs, etc) and decompose into data requirements
3. Consolidate different data requirements into a general [interchangeable fund data format (Github Issue #7)](https://github.com/melonproject/reporting-thesis/issues/7)
4. Map available data to interchangeable fund data format.

This is the old world.

# SFAMA FINMA KIID Template (German)

The first document we look at is the SFAMA [KIID](#_bookmark75) guidelines (in German) which is available on their website and acts as guideline how to compose a [KIID](#_bookmark75) to submit to [FINMA](#_bookmark72). It also contains a [KIID](#_bookmark75) template on pages 13-14, [2].

## Praeamble

Prepended to every [KIID](#_bookmark75) document there is the following legal text:

*Wesentliche Informationen für die Anlegerinnen und Anleger*: Gegenstand dieses Dokuments sind wesentliche Informationen für die Anlegerinnen und Anleger über diese kollektive Kapitalanlage. Es handelt sich nicht um Werbematerial. Diese Informationen sind gesetzlich vorgeschrieben, um Ihnen die Wesensart dieser kollektiven Kapitalanlage und die Risiken einer Anlage zu erläutern. Wir raten Ihnen zur Lektüre dieses Dokuments, sodass Sie eine fundierte Anlageentscheidung treffen können.

Which seems to be the same as in english [KIID](#_bookmark75) documents from [EFAMA](#_bookmark67). Here in english:

*Key information for investors*: This document provides key investor information about this collective investment scheme. It is not marketing material. The information is required by law to help investors understand the nature and the risks of investing in this Fund. Investors are advised to read it so to make an informed decision about whether to invest.

**Data classification**: off-chain, free text, given

## Fund name

123 Kollektive Kapitalanlage [, ein Teilvermögen des 123 Umbrella Fonds, Klasse A] The long name of the fund. E.g. "Credit Suisse (CH) Small [Cap](#_bookmark65) Switzerland Equity Fund" **Data classification**: on-chain, free text, static

## ID/ISIN

([ISIN](#_bookmark74): xxxxyyyyyzzz)

The funds [ISIN](#_bookmark74) identification. For us this can be the address of the deployed fund contract if fund is not submitted to [isin](#_bookmark74).org.

**Data classification**: on-chain, generated

## Legal entity

Fondsleitung: ABC Fondsleitung (Schweiz) AG [, eine Gesellschaft ABC Gruppe]

**Data classification**: off-chain, free text, static

## Objectives and investment policy

Anlageziele und Anlagepolitik

### Asset categories

Hauptkategorien der für die Anlage in Frage kommenden Finanzinstrumente;

A Melon fund selects its manageable assets during setup and they can't change anymore later. So this can be displayed as a simple list.

**Data classification**: on-chain, static, configuration

### Participation

Hinweis auf das Kündigungsrecht des Anlegers unter Angabe der Rücknahmefrequenz

Generally, Melon funds do not restrict investors from withdrawing funds or termination. But every fund can have a separate [Participation](#_bookmark77)/compliance module (see [Melon smart-contracts: ComplianceInterface](https://github.com/melonproject/smart-contracts/blob/v0.7.2-alpha.1/src/compliance/ComplianceInterface.sol#L6)) which can restrict

investing & redemption through boolean functions with the following inputs: , , . Therefore [participation](#_bookmark77) modules can restrict withdrawal according to liquidity, ...

investor

shareQuantity

giveQuantity

The redemption of shares is also restricted if the fund does not hold enough of the quote asset (i.e. cash, e.g. ETH or [DAI](#_bookmark66)). That said, it is always possible to redeem in slices (emergency redeem: Receive the managed assets directly in proportion of shares).

**Data classification**: on-chain, static, configuration

**Data representation**: Common [participation](#_bookmark77) modules could be described with free text.

### Strategy/Objectives

Angabe, ob die kollektive Kapitalanlage ein bestimmtes Ziel in Bezug auf einen branchen-spezifischen, geografischen oder anderen Marktsektor oder in Bezug auf spezifische Anlageklassen verfolgt;

This is just free text describing the strategy/objectives of the fund. For a manual audit, it should be easy to verify if the objectives are followed or not.

**Data classification**: off-chain, free text

### Open questions

What are examples of strategy/objectives? How are those currently enforced?

What happens if a fund manager violates these objectives?

Example: Credit Suisse (CH) Small [Cap](#_bookmark65) Switzerland Equity Fund: Invests in small titles in Switzerland.

What happens if the fund invests in a company not listed in Switzerland?

What happens if a company that this fund is invested in is bought by an US company?

### Automation

Angabe, ob die kollektive Kapitalanlage die Anlageentscheide nach freiem Ermessen treffen kann und ob ein Referenzwert herangezogen wird;

*Note*: It also talks about benchmarks or reference prices but we will go into this topic later.

A Melon fund could be automatically managed by a bot similar to an [ETF](#_bookmark71) which is currently not visible on-chain. In the future it might be possible to store some strategies on-chain but for this thesis we stick to the status quo: A fund can indicate automation just by describing it with free text. An auditor might check a fund for automation which did not declare its automation or check if automation was executed as described.

**Data classification**: off-chain, free text.

### Income distribution

Angabe, ob die Erträge der kollektiven Kapitalanlage ausgeschüttet oder thesauriert werden.

Not applicable. An investor always holds a percentage of the [AUM](#_bookmark64) in Melon. There is no concept of dividends or similar.

**Data classification**: Not applicable

### Leverage, Hedging, Arbitrage, Dept Securities

Werden besondere Anlagetechniken verwendet, wie z.B. „hedging“, „arbitrage“ oder „leverage“, so sind deren Auswirkungen auf die Wertentwicklung der kollektiven Kapitalanlage anzugeben; Investiert die kollektive Kapitalanlage in Schuldtitel, sind diese zu beschreiben;

The Melon smart contracts by itself do currently not actively support leverage, hedging, etc. That said, it is theoretically possible to buy into an asset/token which could be leveraged. E.g. Lendroid, Salt lending, ...

Risk assessment is done per asset.

**Data classification**: Not applicable, free text

### Asset selection criteria

Nach welchen Kriterien die Anlagen ausgewählt werden, z.B. nach „Wachstum“, „Wert“ oder „hohe Dividenden“ mit Erläuterung dieser Kriterien;

In the current smart contract implementation, the asset universe is predefined by the PriceFeed: A fund cannot invest in an asset which is not prelisted on the general pricefeed. Assets are not categorized.

**Data classification**: N/A yet, free text

### Open questions

Will assets on the blockchain be categorized in the future so that a fund can set a asset selection critera as smart contract? E.g. A fund that only invests in renewable energy cannot buy a token representing a share on a atomic power plant.

### Factors influencing performance

bei strukturierten kollektiven Kapitalanlagen die Faktoren, welche die Ausschüttungen sowie die Wertentwicklung der kollektiven Kapitalanlage beeinflussen;

There are no other factors influencing performance than the actual performance of the fund.

**Data classification**: Not applicable, given

### Transaction costs

dass die Transaktionskosten zu Lasten des Fondsvermögens gehen und somit den Ertrag der kollektiven Kapitalanlage schmälern;

Transaction costs (i.e. gas) are paid by the fund manager in the current implementation. It is her responsibility to set the fees in a way that the transaction costs are paid. In future versions we might implement a feature which deducts transaction costs directly from the fund performance.

**Data classification**: Not applicable, given

### Minimum holding period

Sofern im Prospekt oder anderweitig eine Mindesthaltedauer empfohlen wird, ist der folgende Wortlaut beizufügen: „Empfehlung: Diese kollektive Kapitalanlage ist unter Umständen für Anlegerinnen und Anleger nicht geeignet, die ihr Geld innerhalb eines Zeitraumes von […] aus der kollektiven Kapitalanlage wieder zurückziehen wollen.“;

There is no concept of minimum holding period in the current implementation. That said, it might be part of the managers strategy to recommend something or enforces it through a [participation](#_bookmark77) module.

**Data classification**: free-text, (on-chain, static, configuration through a [participation](#_bookmark77) module)

### Funds of funds

Bei Dachfonds ist anzugeben, nach welchen Kriterien die Zielfonds ausgewählt werden.

It is currently forbidden to create fund of funds.

**Data classification**: Not applicable

## Risk profile

### Risk indicator

Comparable indicator between 1 (low risk/low reward) to 7 (high risk/high reward). The SFAMA document [2], pages 16ff describes how to calculate this indicator. It is the same as [ESMA](#_bookmark70)'s [SRRI](#_bookmark82) calculation. It is mainly based on overall volatility of a fund.

**Data classification**: Complicated, historic, on-chain

### Given explanations

There are some given explanations:

* + dass der Indikator keine zuverlässige Aussage über die zukünftige Wertentwicklung enthält; • dass die Risikokategorie Veränderungen unterliegen und über die Jahre variieren kann; • dass die geringste Risikokategorie nicht einer risikofreien Anlage entspricht;

Which are just warnings about risks in general.

**Data classification**: Given

# Timely reports

**MiFID II and PRIIP**

**Melon Risk Engineering**

**uPort**

**Solidity research**

## Resources

### Solidity

[Solidity](https://solidity.readthedocs.io/) is a programming language for smart contract development on the Ethereum blockchain.

### Parity

[Parity](https://www.parity.io/) is a client to interact with the Ethereum blockchain. It provides an easy way to set up a development chain where we can test our own smart contracts without limitations.

### Web3.js

[Web3.js](https://web3js.readthedocs.io/en/1.0/index.html) is a JavaScript API for interacting with local or remote ethereum nodes. It enables us to develop clients which communicate with our smart contracts on the blockchain.

## Learnings

While learning [*Solidity*](#_bookmark81), we created a cheatsheet to have a compact summary while developing our own smart contracts. The cheatsheet can be found in the appendix Solidity cheatsheet.

The [*ABI*](#_bookmark62) (Application Binary Interface) can be understood as the API to a deployed smart contract. *Web3.js* or a similar library uses this interface to interact with the Ethereum ecosystem.

*Web3* requires us to specify a provider. Here we point to a running Ethereum client, in our case a running *Parity* instance. Through the [ABI](#_bookmark62), we can call functions on the contract instance through contract.methods . Methods that store value on the blockchain can be executed by send() , Read-only methods can be called with call() . The result

is a JSON RPC response containing the requested values. It is possible to catch the or

transactionHash

error ,

receipt with the web3-function on() . All events on the chain can be retrieved with the getPastEvents() method. We could also subsrcibe on new events and get notified instantly when they are fired from withing smart contracts.

## Assessment

The *melon protocol* is already dependent on the Ethereum blockchain and uses smart contracts written in [Solidity](#_bookmark81) extensively. Furthermore, smart contracts support us to store melon fund audit information in a secure and consistent way and provide us with the features we require to verify auditors.

We also looked into [truffle suite](http://truffleframework.com/) and [dapp tools](https://dapp.tools/). *Parity* and *web3.js* provide us with all the features we need to develop smart contracts and interact with them for now though.

## Possible limitations and considerations

It is of upmost importance that the smart contracts we develop for this project are as light as possible. Storing data on the blockchain can be expensive. We presume that it is in the interest of the auditors that the fee of storing audit information of a fund is as small as possible. For this, we will keep an eye on the *gas cost* that is required to store information.

As this project and the corresponding smart contracts are not dealing with *Ether* directly, we do not have to account security considerations as in sending or receiving currencies. Most of the data that we will be acquiring from the blockchain can be regarded as *read only*.

# Solidity Cheat Sheet

This is a summary of [Solidity In Depth](https://solidity.readthedocs.io/en/develop/solidity-in-depth.html).

# Layout of source files

## Versions

pragma solidity ^0.4.0;

^ means "only works with compilers 0.4.x"

We will use at least version *0.4.20* for our contracts.

## Import

import "filename"; // import from 'global' or same directory import \* as symbolName from "filename";

import "filename" as symbolName; // same as above import {symbol1 as alias, symbol2} from "filename";

import "./x" as x; // strictly import from same directory

**Path prefix remapping**

If we clone github.com/ethereum/dapp-bin/ locally to /usr/local/dapp-bin, we can use:

import "github.com/ethereum/dapp-bin/library/iterable\_mapping.sol" as it\_mapping;

And run the compiler with:

solc github.com/ethereum/dapp-bin/=/usr/local/dapp-bin/ source.sol

## Comments

// single line

/\*

Multiline

\*/

**Natspec comments**

/// single line

/\*\* multiline \*/

Example:

/\*\* @title Shape calculator. \*/ contract shapeCalculator {

/\*\* @dev Calculates a rectangle's surface and perimeter.

* @param w Width of the rectangle.
* @param h Height of the rectangle.
* @return s The calculated surface.
* @return p The calculated perimeter.

\*/

function rectangle(uint w, uint h) returns (uint s, uint p) { s = w \* h;

p = 2 \* (w + h);

}

}

# Structure of a contract

## State Variables

Permanently stored in contract storage.

unit storedData;

## Functions

function bid() {...}

**Function modifiers**

Amend the semantics of a function, mostly used for require. Declaration:

modifier onlySeller() { ... }

Usage:

function abort() onlySeller { ... }

## Events

Interfaces for EVM logging Declaration:

event HighestBidIncreased(address bidder, uint amount);

Trigger:

emit HighestBidIncreased(msg.sender, msg.value);

## Struct Types

Group several variables

struct Voter { uint weight; bool voted;

address delegate; uint vote;

}

## Enum Types

enum State { Created, Locked, Inactive }

**Types**

**Value Types**

### Booleans

bool t = true; bool f = false;

Operators: !, &&, ||, ==, !=

### Integers

Aliases for int256 and uint256:

int i = -1; uint j = 1;

Declare size from (u)int8 to (u)int256 , e.g.:

int24 max = 8388608;

Comparisons: <=, <, ==, !=, >=, > Bit operators: &, |, ^, ~ Arithmetic: +, -, *, /, %, \**, <<, >>

### Fixed point numbers

They are not fully supported in [Solidity](#_bookmark81) yet, they can only be declared.

### Address

20 byte value. Operators: <=, <, ==, !=, >=, >

### Members:

Query the **balance** of an address:

uint b = a.balance;

**Send ether** in units of wei to an address:

a.transfer(10); // transfers 10 wei to address a

If the execution fails, the contract will stop with an exception If a is a contract address, its code will be executed with the transfer call.

Low level counterpart of transfer (returns 'false' on fail):

bool success = a.send(10);

*NOTE*: send is dangerous, use transfer or the withdraw pattern.

**call, callcode, delegatecall**: interface with contracts that do not adhere with the [ABI](#_bookmark62).

*NOTE*: Only use as last resort, they break the type-safety of [Solidity](#_bookmark81). Arguments of call are padded to bytes32 type. Returns bool that indicates if the function terminated (true) or threw an exception (false).

address nameReg = 0x72ba7d8e73fe8eb666ea66babc8116a41bfb10e2; nameReg.call("register", "MyName"); nameReg.call(bytes4(keccak256("fun(uint256)")), a); // function signature

Adjust the supplied gas with .gas():

namReg.call.gas(1000000)("register", "MyName");

Control the supplied ether value:

nameReg.call.value(1 ether)("register", "MyName");

Combine both:

nameReg.call.gas(1000000).value(1 ether)("register", "MyName");

Query the **balance of the current contract**:

this.balance;

### Fixed-size byte arrays

, up to bytes32

|  |  |  |
| --- | --- | --- |
| bytes1 , | bytes2 , | bytes3 |

bytes is an alias for bytes1 .

Comparisons and bit operators can be used like on ints.

##### Index access:

bytes2 b2 = "hi";

byte b1 = b2[0]; // access first byte, read only!

**Length**:

b1.length; // returns the fixed length of the byte array

bytes and string are not value types!

### Rational and integer literals

##### Decimal fraction literals:

1.

.1

1.3

**Scientific notation** is supported:

2e10

-2e10 2e-10

2.5e1

Division on integers converts to a rational number, but it cannot be stored in any way yet.

int i = 5 / 2; // this throws a compiler error int j = 6 / 2; // this works

### String literals

"use double quotes" 'or single quotes'

They are implicitly convertible to bytes1 ... bytes32 **if they fit**. **Escape characters** are possible:

\n // and the like

\xNN // hex

\uNNNN // unicode

### Hexadecimal literals

hex"001122FF"

The content must be a hexadecimal string. The value will be the binary representation.. They behave like string literals.

### Enums

enum ActionChoices { GoLeft, GoRight, GoStraight, SitStill }

ActionChoices choice = ActionChoices.GoStraight;

**Function types**

##### internal functions

Can only be called inside the current contract. This is the *default* for functions. Use internal functions in **libraries**:

library SomeLibrary {

// declare internal functions

}

contract SomeContract { using SomeLibrary for \*;

// use internal functions

}

##### external functions

Consist of an address and a function signature. Use external functions between **contracts**:

contract Oracle {

// define external functions

}

contract OracleUser {

Oracle constant oracle = Oracle(0x1234567); // known contract oracle.query(...);

}

##### Notation:

function (<parameter types>) {internal|external} [pure|constant|view|payable] [returns (<return types>)]

Access the function type:

f // call by name -> internal function this.f // -> external function

Return the [ABI](#_bookmark62) function selector:

this.f.selector; // type: bytes4

## Reference Types

Reference types have to be handled more carefully, because storing them in **storage** is expensive.

### Data location

Complex types (arrays & structs) have a data location ( storage or memory ).

Default for function parameters is memory

Default for local variables is storage

storage is forced for state variables

There is a third location calldata where function arguments are stored.

### Arrays

Can have a fixed size or can be dynamic. Array of 5 dynamic arrays of uint:

uint[][5] a;

uint a32 = a[2][3]; // access second uint in third array

**NOTE**: notation is reversed!!!

**USEFUL** - convert from string to bytes:

string memory s = "hello world"; b = bytes(s);

Create a **getter** for arrays automatically by marking them public :

int32 public intArray;

But values can only be obtained with a numeric index.

### Allocating Memory Arrays

Use new for arrays with variable length in *memory*.

uint[] memory a = new uint[](7); bytes memory b = new bytes(7);

### Array Literals / Inline Arrays

Arrays written as an expression:

[uint(1), 2, 3]; // evaluates to a 'uint8[3] memory' array

The cast on the first element is necessary to define the type. They cannot be assigned to dynamic arrays at the moment.

### Members

length

*Dynamic* arrays can be resized in *storage* with .length . The size of *memory* arrays is fixed once they are created.

push

Use push to append an element on dynamic storage arrays and bytes .

##### NOTE:

It is not possible to return dynamic content from external function calls. The only workaround now is to use large statically-sized arrays.

### Structs

struct Funder { address addr; uint amount;

}

Funder f1 = Funder({addr: msg.sender, amount: msg.value}); // create with names Funder f2 = Funder(msg.sender, msg.value); // simple create

f1.amount += msg.value; // access

**Mappings**

Formal definition:

mapping(\_KeyType => \_ValueType)

Keytype can be almost anything except for a mapping, dynamically sized array, contract, enum or struct. ValueType can be anything, even another mapping.

The key data is not actually stored in a mapping, only its keccak256 hash. Mappings do not have a length. Mappings are only allowed for **state variables** (or storage reference types in internal functions).

The only way to retreive a value from a mapping is by its key. Mappings are not enumerable. Example:

// declaration

mapping(address => Voter) public voters;

// change a value in a mapping chairperson = msg.sender; voters[chairperson].weight = 1;

## LValue Operators

a += e; a -= e; a \*= e; a /= e; a %= e; a |= e; a &= e; a ^= e; a++;

a--;

++a;

--a;

### delete

delete assigns the initial value for the type of a:

int i = 42;

delete i; // i is now 0

Delete on dynamic arrays assigns an array of size 0. Delete on static arrays resets all values.

Delete on structs resets all values.

## Conversions

### Implicit conversions

Possible when no information is lost:

uint8 to uint17 is possible

int128 to int256 is possible

int8 to uint256 is NOT possible

uints can be converted to bytes of the same size or larger

uint160 can be converted to address

### Explicit conversions

Use with care!

int8 y = -3; uint x = uint(y);

uint32 a = 0x12345678;

uint16 b = uint16(a); // b will be 0x5678 now -> information loss

## Type Deduction (var)

It is not necessary everytime to assign a type.

uint24 x = 0x123;

var y = x; // y has type uint24 automatically here

# Units and global variables

## Ether units

Possible ways to work with ether units:

Literal number with suffix ( , , , )

wei

finney

szabo

ether

Literal number without suffix is always wei

Calculation works as expected.

2 ether == 2000 finney // evaluates to true

## Time units

Calculation works as expected. The base unit is *seconds*.

1 == 1 seconds

1 minutes == 60 seconds

1 hours == 60 minutes

1 days == 24 hours

1 weeks == 7 days

1 years == 365 days

The suffixes cannot be applied to variables. Do it like this:

uint daysAfter = 42;

if (now >= daysAfter \* 1 days) {...};

## Special variables and functions

### Block and transaction properties

block.blockhash(uint blockNumber) returns ( bytes32 ): hash of the given block - only works for 256 most recent blocks excluding current

): current block miner’s address

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| block.coinbase ( | | address | | | | |
| block.difficulty | | | ( uint | | |  |
| block.gaslimit ( | | uint ) | | | : c | |
| block.number | ( uint | | | ): cur | | |
| block.timestamp | | ( | uint | | ): | |

): current block difficulty urrent block gaslimit rent block number

current block timestamp as seconds since unix epoch

gasleft() returns ( uint256 ): remaining gas

): complete calldata

( bytes

msg.data

): sender of the message (current call)

address

(

msg.sender

): first four bytes of the calldata (i.e. function identifier)

bytes4

msg.sig (

): number of wei sent with the message

( uint

msg.value

): current block timestamp (alias for block.timestamp)

( uint

now

( ): gas price of the transaction

uint

tx.gasprice

: sender of the transaction (full call chain)

( address )

tx.origin

**NOTE:** now is just the timestamp from the block!

### Error handling

assert(bool condition) // for internal errors

require(bool condition) // for errors in inputs or external components revert() // abort execution, revert state changes

**Mathematical and cryptographic functions**

addmod(uint x, uint y, uint k) returns (uint) // compute (x + y) % k mulmod(uint x, uint y, uint k) returns (uint): // compute (x \* y) % k keccak256(...) returns (bytes32) // compute Ethereum-SHA-3 hash of args sha256(...) returns (bytes32) // compute SHA-256 hash of args sha3(...) returns (bytes32) // alias to keccak256()

ripemd160(...) returns (bytes20) // compute RIPEMD-160 hash of args

ecrecover(bytes32 hash, uint8 v, bytes32 r, bytes32 s) returns (address) // elliptic curve signature

The arguments are packed without padding. [ecrecover example](https://ethereum.stackexchange.com/questions/1777/workflow-on-signing-a-string-with-private-key-followed-by-signature-verificatio)

### Address related

<address>.balance (uint256) // balance of the address in wei

<address>.transfer(uint256 amount) // send amount of wei to address, throws on failure

<address>.send(uint256 amount) returns (bool) // send amount of wei to address, returns false on failure

**Contract related**

this // the current contract, convertible to address

selfdestruct(address recipient) // destroy current contract, send funds to address

**Expressions and control structures**

**Input parameters**

function taker(uint \_a, uint \_b) public pure {

// do something with \_a and \_b.

}

**Output parameters**

Returning multiple values is possible.

function arithmetics(uint \_a, uint \_b) public

pure

returns (uint a, uint b)

{

a = 1;

b = 2;

}

Return parameters are initialized to zero.

## Control structures

They can be used the same as in C or Javascript:

if

#### 

else

while

do

for

break

continue

return

? :

There is no type conversion from non-boolean to boolean ( 1 is not true !).

## Function calls

### Internal function calls

Internal functions (from the same contract) can be used recursively.

### External function calls

this.g(8);

c.g(8) // where c is a contract instance

Amount of wei and gas can be specified when calling functions from other contracts:

contract InfoFeed {

function info() public payable returns (uint ret) { return 42; }

}

contract Consumer {

InfoFeed feed; // contract instance

function callFeed() public { feed.info.value(10).gas(800)(); }

}

payable must be used for info() to have the .value() option.

##### WARNING:

Called contracts can change state from our own contracts. Write functions in a way that calls to external functions happen after any changes to state variables in our contract so our contract is not vulnerable to a *reentrancy exploit*.

### Named calls

Enclose args in {} , then the order doesn't matter:

f({value: 2, key: 3});

## Creating contracts with new & constructors

Contracts can create other contracts with new :

contract D {

uint x;

function D(uint a) public payable { // ctor x = a;

}

}

contract C {

D d = new D(4); // will be executed as part of C's constructor

function createD(uint arg) public { D newD = new D(arg);

}

}

**NOTE**: Creating constructors by their name is deprecated. This is the new, preferred way:

constructor() public payable {

//

}

## Assignment

Tuple syntax is possible:

function f() public pure returns (uint, bool, uint) { return (7, true, 2);

}

var (x, b, y) = f(); // specifying types is not possible here (x, y) = (2, 7); // assign to predefined variables

(x, y) = (y, x); // swap

(data.length,) = f(); // rest of the values can be ignored (returns 2 values but we only care about first) (,data[3]) = f(); // ignore beginning values

(x,) = (1,); // one component tuple

## Scoping and declarations

For version 0.4.x, a variable declared anywhere in a function is available everywhere in the function (like Javascript). In version 0.5.x, this will change.

## Error handling (Assert, Require, Revert, Exceptions)

[Solidity](#_bookmark81) uses state-reverting exceptions.

Use assert to check invariants Use require to ensure input values

Use revert to throw an exception, revert the current call (and done subcalls) Catching exceptions is not yet possible.

assert will use all gas, require uses none.

# Contracts

## Creating Contracts

With web3js: [web3.eth.Contract](https://web3js.readthedocs.io/en/1.0/web3-eth-contract.html#new-contract)

Only one constructor is allowed --> ctor overloading is not possible. Cyclic dependencies between contracts are not possible.

## Visibility and getters

There are two kinds of function calls in [Solidity](#_bookmark81), so there are four types of **visibilities for functions**.

### external

Called via other contracts and transactions.

To call it from inside a contract, we have to use this.f() .

### public (default)

Call internally or via messages. For public state variables, a getter is generated automatically.

### internal

Functions and state variables can only be accessed from within the *contract (or deriving contracts)*.

### private

Functions and state variables can only be accessed from within the *contract*.

**NOTE:** private stuff is still visible for everyone, just not accessible!

## Getter functions

The compiler automatically creates getter-functions for public state variables. This:

contract Complex { struct Data {

uint a; bytes3 b;

mapping (uint => uint) map;

}

mapping (uint => mapping(bool => Data[])) public data;

}

will generate the following function:

function data(uint arg1, bool arg2, uint arg3) public returns (uint a, bytes3 b) { a = data[arg1][arg2][arg3].a;

b = data[arg1][arg2][arg3].b;

}

## Function modifiers

Modifiers can change the behaviour of functions. The function body is inserted where the special symbol **\_** appears.

modifier onlyOwner { require(msg.sender == owner);

\_;

}

function changePrice(uint \_price) public onlyOwner { price = \_price;

}

Multiple modifiers can be used in a whitespace-separated list. All symbols visible in the function are visible for the modifier.

## Constant state variables

State variables can be declared as constant . Then they have to be assigned from an expression which is a constant at compile time.

These functions are allowed:

|  |  |
| --- | --- |
| keccak256 | |
| sha256 |  |
| ripemd160 | |
| ecrecover | |
| addmod |  |
| mulmod |

The only supported types valid for now are **value types** and **strings**.

## Funtions

### View functions

Promise **not to modify state**. Can be declared with view . These things are considered to modify state:

Writing to state variables Emitting events

Creating other contracts Using selfdestruct Sending ether

Calling functions not marked view or pure

Using low-level calls

Using inline assembly with opcodes

**NOTE:** getter methods are marked view .

### Pure functions

Functions that do **not read from or modify the state**. These things are considered to read from state:

Reading from state variables

Accessing this.balance or <address>.balance

Accessing or msg

tx

block ,

Calling functions not marked with pure

Inline assembly with opcodes

### Fallback function

The unnamed function. This is called when no other function matches the function identifier. Sending ether to this contract will cause an exception (no other functions are defined):

uint x;

function() public { x = 1; }

If ether is sent to this contract, there is no way to get it back:

function() public payable { }

### Function overloading

Function overloading is possible (but not for ctors).

contract A {

function f(uint \_in) public pure returns (uint out) { out = 1;

}

function f(uint \_in, bytes32 \_key) public pure returns (uint out) { out = 2;

}

}

If there is not exactly one candidate for the function, resolution fails. For example: f(unit8) and f(uint256) fails when f is called with a uint8 value or below.

## Events

The "logging" mechanism of ethereum.

SPV proofs are possible: If an external entity supplies a contract with an SPV proof, it can check that the log actually exists in the blockchain (but block headers have to be supplied).

Up to three arguments can receive the attribute indexed . We can then search for these arguments. Indexed arguments will not be stored themselves, we can only search for these values. If arrays are used as indexed arguments, their keccak256 hash will be stored.

With anonymous , the signature of the event is not stored. All non-indexed arguments will be stored in the data part of the log.

Event example:

contract ClientReceipt { event Deposit(

address indexed \_from, bytes32 indexed \_id, uint \_value

);

function deposit(bytes32 \_id) public payable {

emit Deposit(msg.sender, \_id, msg.value); // 'Deposit' is now filterable with JS

}

}

Look for events with *javascript*:

var abi = /\* abi as generated by the compiler \*/; var ClientReceipt = web3.eth.contract(abi);

var clientReceipt = ClientReceipt.at("0x1234...ab67" /\* address \*/); var event = clientReceipt.Deposit();

// watch for changes event.watch(function(error, result){

// result will contain various information

// including the argumets given to the `Deposit`

// call. if (!error)

console.log(result);

});

// or callback to start watching immediately

var event = clientReceipt.Deposit(function(error, result) { if (!error)

console.log(result);

});

There is also a **low-level interface** for logs.

## Inheritance

[Solidity](#_bookmark81) supports multiple inheritance. All functions are *virtual* (most derived function is called). The code from inherited contracts is copied into one contract.

Use is to derive from another contract.

If a contract doesn't implement all functions, it can only be used as an interface:

function lookup(uint id) public returns (address adr); // 'abstract' function

Multiple inheritance is possible:

contract named is owned, mortal { ... }

Functions can be overridden by another function with the same name and the same number/types of inputs. If the constructor takes an argument, it must be provided like this:

contract PriceFeed is named("GoldFeed") { ... }

To specifically access functions from base contracts, use super :

contract Base is mortal {

function kill() public { super.kill(); }

}

### Constructors

Constructor functions can be public or internal .

contract B is A(1) { function B() public {}

}

An internal ctor marks the contract as abstract!

### Arguments for base constructors

contract Base { uint x;

function Base(uint \_x) public { x = \_x; }

}

contract Derived is Base(7) {

function Derived(uint \_y) Base(\_y \* \_y) public {

}

}

**Abstract contracts**

Contracts where at least one function is not implemented. This is a function declaration:

function foo(address) external returns (address);

Careful: this is a function type (variable which type is a function):

function(address) external returns (address) foo;

## Interfaces

Cannot have any functions implemented Cannot inherit other contracts or interfaces Cannot define variables

Cannot define structs Cannot define enums

Use keyword interface :

interface Token {

function transfer(address recipient, uint amount) public;

}

## Libraries

Libraries are assumed to be stateless. They are deployed only once at a specific address. Restrictions in comparison to contracts

No state variables

Cannot inherit or be inherited Cannot receive ether

Example:

library Set {

// type of Data is 'storage reference', so only the storage address, not the content is saved here struct Data { mapping(uint => bool) flags; } // will be used in the calling contract!

function insert(Data storage self, uint value) public

returns (bool)

{

if (self.flags[value])

return false; // already there self.flags[value] = true;

return true;

}

// ...

contract C {

Set.Data knownValues;

function register(uint value) public {

// library functions can be called without a specific instance!

// the 'instance' is the current contract... require(Set.insert(knownValues, value));

}

## Using for

Attach library functions to any type:

using A for B; // where A is the library and B the type

With the example from above:

contract C {

using Set for Set.Data; // this is the crucial change Set.Data knownValues;

function register(uint value) public {

// Here, all variables of type Set.Data have

// corresponding member functions.

// The following function call is identical to

// `Set.insert(knownValues, value)` require(knownValues.insert(value));

}

}

We can also extend elementary types:

using Search for uint[];

# Data Classification

## Reasoning

### CAP theorem

The Ethereum blockchain does not behave like a typical distributed data store in the classical sense. Nonetheless, we want to discuss the [CAP](#_bookmark65) theorem and its guarantees here.

Source: [Wikipedia](https://en.wikipedia.org/wiki/CAP_theorem)

### Consistency

Every read receives the most recent write or an error

Regarding On-Chain data for the reports, this is not relevant. We generate reports in regards to a timespan. This means that we do not have to depend on values being the "most recent".

### Availability

Every request receives a (non-error) response – without guarantee that it contains the most recent write Again - with On-Chain data, we work with data retrieved from a timespan. Availability is not an issue.

### Partition tolerance

The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes

As every node in the blockchain eventually works with the same ledger, partition tolerance is not an issue.

### ACID

**Atomicity Consistency Isolation Durability**

**Eventual consistency**

[BASE (Basically Available, Soft state, Eventual consistency)](https://en.wikipedia.org/wiki/Eventual_consistency)

[The Ethereum blockchain does NOT follow *eventual consistency*, but *strong consistency*. Source: Bitcoin Guarantees Strong, not Eventual, Consistency](http://hackingdistributed.com/2016/03/01/bitcoin-guarantees-strong-not-eventual-consistency/)

### Our approach

**Call for collaborators**

**Summary**

Melonport creates a set of smart contracts that facilitates crypto asset management on the blockchain. There will be traditionally regulated "crypto funds" that build upon these. This thesis engineers the reporting and auditing of such funds.

We are looking for collaborators who are willing to shape the future of reporting and auditing of such "crypto funds".

Attached to this brief is a mockup to illustrate how a Melon fund report will look like which also can be downloaded [here](https://melon-reporting.now.sh/assets/Melon-Report-Mockup-v2.pdf).

Furthermore, there is a short refresher about blockchain/smart-contracts/ melon-protocol at the end and links to in- deep topics from this thesis.

## The case for reporting / auditing

Melonport is already talking to people who want to setup and manage crypto funds powered by the Melon protocol fully regulated. In order to be legally compliant they need to incorporate a solid reporting and auditing process. Some aspects of this process differ from the traditional process. Namely data sourcing: One can extract almost all relevant data reliably directly from the immutable blockchain:

Investments: Every invest/redeem into the fund is directly linked to an address and through the transaction hash to the exact block.

Trades: It is possible to retrace every trade of the fund manager, also with a verifiable distinct transaction hash. Policy: A fund is setup with certain policies which are immutable and automatically enforced like KYC, fees, asset whitelist, best price exection, ... See [Risk Engineering](http://www.docs.melonport.com/chapters/risk_engineering.html#risk-engineering)

Identification: It's possible to resolve some addresses to persons through projects like uPort. Audits: The span of audits and who performed them with which opinion.

In this thesis we create:

A data extractor to extract this data from the blockchain A data schema that describes the format of this data

A data visualiser to make this raw data easily readable, understandable and thus verifiable A data signer to cryptographically sign & approve a certain set of data

## Call to action

In order to test the assumptions made in this thesis we need a knowledgeable sparring partner from the industry who understands:

The blockchain The Melon protocol

Its implications on the financial sector in general

Its implications on reporting and auditing in specific Someone who can help us to answer the following question:

Is a report like the one attached sufficient?

If something is missing: What?

Does the data schema makes sense like this?

What could be the focus of an auditor of blockchain funds? Where are the weaknesses that needs to be audited?

## Refresher

### Blockchain (Ethereum)

The blockchain is a decentralized and trustless computer with with a guaranteed transaction history that cannot be tampered with.

It is possible to write smart contracts which are nothing more than just code consisting of:

A data schema with an initial state Rules to change this state

E.g. An ERC20 token is just a registry of token holders with their balances and rules how they can "transfer" these tokens or change the balances.

Smart contracts running on the same blockchain can interact with each other.

Users are identified through a public-key/private-key cryptographic mechanisms.

### Melon Protocol

The Melon protocol is a set of smart contracts on the Ethereum blockchain that enables setting up and managing a fund consisting of ERC20 tokens.

Technically, a Melon fund is more like a managed account.

It has two main actors:

Manager:

Sets up the fund: Defining the rules like: Allowed exchanges

Performance fee Management fee Best price execution

Asset (Token) white-list

...

Manages the fund:

Executes trades on decentralised exchanges in the name of the fund

Investor:

Invests into a fund and receives shares for it

Can redeem at any time and receives the tokens according to his shares Under the following guarantees:

Rules set at fund setup cannot be changed later

The fund manager cannot embezzle the assets in the fund

## Links

### General Melon

Current state of the interface: <https://ipfs.io/ipns/melon.fund/> General Melon Protocol docs: <http://www.docs.melonport.com/>

### Reporting & Auditing

Use cases: <https://melon-reporting.now.sh/01-thesis/04-analysis/01-UseCases.html>

On-chain data [ERM](#_bookmark69): <https://melon-reporting.now.sh/01-thesis/04-analysis/03-OnChainData.html> System overview: <https://melon-reporting.now.sh/01-thesis/05-solution/01-SystemOverview.html> [Fund Report Data Schema: https://melon-reporting.now.sh/01-thesis/05-solution/04- FundReportDataSchema.html](https://melon-reporting.now.sh/01-thesis/05-solution/04-FundReportDataSchema.html)

[Auditing contract standard: https://melon-reporting.now.sh/01-thesis/05-solution/05- MelonAuditingContractStandard.html](https://melon-reporting.now.sh/01-thesis/05-solution/05-MelonAuditingContractStandard.html)

# Minutes

## Meeting 2018-07-05

Present:

Simon Emanuel Schmid Benjamin Zumbrunn Sarah Hauser (FHNW) Markus Knecht (FHNW) Mona El Isa (Melonport) John Orthwein (Melonport) Jenna Zenk (Melonport)

### Agenda

Goals Walkthrough Updated Mockup Feedback so far

Mock data generator proposal Data extractor

New contract + tests FHNW misc

### Notes

http://localhost:3000/report/0xC4ce568951958746904065Ee9E1C0aeE927D5260/1530267228/1530720180

### To Discuss:

Investor data is not easily available Only through event logs

Even a simple list of investors is not easily extractable Maybe create helper functions

### Walkthrough

[Also save text representation of "opinion" to show better what's not good. (#118)](https://github.com/melonproject/reporting/issues/118)

### Updated Mockup

Write down the reasoning of the Web Interface in the thesis! Why is it easy to use, why something is implemented this way, Stakeholders agreed on it etc.

Addressed in:

[Extract specification from use cases #131](https://github.com/melonproject/reporting/issues/131)

[Describe & sell "Standard Report Web Interface" #133](https://github.com/melonproject/reporting/issues/133)

### Mock data generator proposal

Arguments for generator:

Fund Database and Blockchain Sync Service will be done by Melonport anyway Mock data could inspire the redesign of the new protocol

Arguments against generator:

Disconnection from "real project"

Build a hybrid with real data (like meta, audits...) and mockup data (trades...) Adressed in: [Test data generator #81](https://github.com/melonproject/reporting/issues/81)

### Data extractor

**New contract + tests**

Argue about worst, mid and best case Big O notation! also visualize this: <https://github.com/melonproject/reporting/issues/134>

Variants expensiveness Make really clear: gas cost and theoretical!

-> [Visualize Worst, Average & Best Case Scenarios for Variant 1 & 2 #135](https://github.com/melonproject/reporting/issues/135)

Maybe argument about building blocks, not variants Show how the architecture looks for the variants... (what is a map?)

Come back to "reality"... how does an auditor usually operate and which variant would be best for this? Argue about risk management!!! isComplete is most important there

Problem with gas limits: are there any limits with filling gaps?

### FHNW misc

Do weekly updates again Focus also on "easy to use interface".

## Meeting 2018-06-12

Present:

Simon Emanuel Schmid Benjamin Zumbrunn Sarah Hauser (FHNW) Markus Knecht (FHNW)

### Agenda

Meeting PWC Final Specification GraphQL

Unit tests

### PWC

Document the "priorities" of PWC --> they are interested that the data shown is correct, they want to know why it is correct, where it comes from etc. Hauser: stay with PWC, otherwise the "initial work" was useless. [#136](https://github.com/melonproject/reporting/issues/136)

### Final specification

Determinism: it's more than just that: it's a "transient function" in functional programming. It's probably *bijective* in mathematics. Maybe it's better to not use one of these terms and just describe what the implications of the extractor function is --> Draw a box in the documentation with the exact specification.

For trusting the data, the database layer gives another problem. Better: think about how we can easily verify the correctness of the data.

Think about the actors and their needs

Auditor: has to trust the data, because he signs it Investor: wants to quickly see the reports

Idea: Just save the report JSON after an audit was made to a database. An investor could then easily look at the report. Plus: Verification is also easy then: compare the datahash of the JSON with the hash from the audit on the blockchain.

But: look also for other ideas! Tradeoffs from verification.

### Smart Contract & storage

Document the risk management stakeholder better: why does he need the isComplete function, etc. [#137](https://github.com/melonproject/reporting/issues/137) Look for other possibilities how to sort the array...

Idea: maybe audit saving with mapping is possible? linked list data structure in [solidity](#_bookmark81)? Idea: maybe say "only 4

audits per year are possible". Then the size / timespans would be fixed. Test long arrays with lots of audits: is the gas cost reasonable?

### Next steps

Implement alpha version (5.7.2018) (super tight schedule) Documentation

## Meeting 2018-06-07

Present:

Simon Emanuel Schmid Benjamin Zumbrunn Michael Taschner (PWC)

Sabine Bartenschlager-Igel (PWC)

### Reporting

Michael: It would be interesting to retroactively construct transactions (with historic prices). [#81](https://github.com/melonproject/reporting/issues/81) Relevant for Melon in general: look at rules of traditional asset management. Sabine & Michael will look into: what is important for an auditor when going through a report.

Note: We used most of the time to explain blockchain and Melon funds, we did not focus on the report.

Note: The people from PWC use the term *managed account*, not *fund* for Melon funds.

## Meeting 2018-05-22

Present:

Simon Emanuel Schmid Benjamin Zumbrunn Markus Knecht (FHNW)

### Documentation

Be careful what is important (appendix/content).

### Expert meeting

Write mail to expert: in which products might he be interested now? (Maybe he does not want to see anything at all for now.)

### Bachelor defence

Maybe hold presentation in english, but this has to be communicated / asked beforehand. Assessment sheet: Knecht sends it to us (with the bonus points for blockchain projects).

### Method

Say explicitly that the approach from Melonport (and in the Blockchain space generally) is something new (old world/new world) and wants to be a bit *disruptive*.

--> Describe Melonport in some sentences. See: [#109](https://github.com/melonproject/reporting/issues/109)

### Modular approach

Try to build systems that are as modular as possible and describe this in the documentation.

### Test presentation

Organize a rehearsal presentation after the thesis contribution date (week 34?).

### Tasks

Write mail for english presentation Write mail to expert

Organize rehearsal presentation

## Meeting 2018-05-18

Present:

Simon Emanuel Schmid

Benjamin Zumbrunn Markus Knecht (FHNW) Jörg Luthiger (FHNW) Konrad Durrer

### System architecture

Arrows are wrong or not very descriptive --> Do this in proper UML

### Questions

What is the database technology? --> look with Melonport

Describe WHY we used the technologies, most importantly for the consolidation service/database Evaluate the

In which interval does the consolidation service cache the data?

Provide a "button" to toggle the consolidation service or use "own" software

"Usability in the technology stack": how easy is it to use the product without the consolidation service?

### Thesis defence

Date: 3.9.2018 Morning

## Meeting 2018-05-03

Present:

Simon Emanuel Schmid Benjamin Zumbrunn Mona El Isa (Melonport) Reto Trinkler (Melonport) John (Melonport) Markus Knecht (FHNW)

### Questions

Volatility for reports: from [inception](#_bookmark73) to timespanEnd or from timespanStart to timespanEnd? Risk Mgmt: is data immutable?

priceHistory: is daily interval good?

priceHistory: which value? mean or median or snapshot? Off-chain data / legal data we need to consider

Is there a possibility to look up names for addresses (uPort)? if yes, are they "hashable"? Datahash could be a direct ipfs hash? (ipfsjs)

### Agenda

Goals

Use Cases Data Schema MIP

Mockup Discussion

Questions

### Goals

Goals and use cases are agreed upon by the coaches and client (i.e. Melonport)

### Use Cases

John: Comment timespan is usually there, but not legally binding (just provides context) Enhance report with costs around funds (gas prices)

### Data Schema

priceHistory: in the old world its a snapshots maybe use "volume weighed average price"? might be too complicated to calculate...

### MIP

What if an auditor refuses an audit? New: status for add() function

unqualified opinion (good) qualified opinion (bad) disclaimer opinion adverse opinion

### Mockup

Strategy might be optional for fund managers, but is highly encouraged!

### Meta

Possible new attributes

Style Substrategies

Substyles --> List of strategies / styles to pick from

John provides us with the most recent strategies and styles from the hedge fund. We might have to invent new strategy/style standards for crypto.

Discussion about whitelist: we now assume that it is fixed. Might be future work ("changing whitelists"):

Sharpe ratio: absolute measure (positions) -> we leave it in. Alpha is not a trivial calculation.

Volatilty: minimum 12 months, if the timespan is < 12 months, it says N/A (Mona provides example for <12 years) Redeem fee might come in the future

Benchmark is not in the fund report data yet. Benchmarks have to be bound to strategy

Weighted average of the share price of all fund weighed by [AUM](#_bookmark64). Proposal from John: not benchmark, but **reference fund**

### Holdings

Token correlation matrix is good

correlation between -1 and top right values are all 1 max drawdown

downside deviation top 3 gainers / losers price change

### Trades

for every trade: show gas price filters (for the website) transaction hash of trade

Profit (for profitability), is an absolute number New Chapter/Page: Positions

Open/Close: Mona gives a few examples

### Participation

Knecht looks up if names can be looked up with ENS

show addresses with green tick

... (kind of off-topic) can the manager "kick" red flag participants?

### Audits

ENS names for auditors (it's about reputation for them)

### Discussion

one real stakeholder for each actor (Mona: maybe regulator from Liechtenstein?) include Bank Frick for feedback

Use cases: show which value from the report is useful for which actor

### Questions

Possible Off-chain data / legal data:

Jurisdiction

Don't waste time on name lookup.

Sha256 should fit into bytes32, just save the hex representation.

Knecht: think about extensibility of the product (so future work is easily implementable).

## Coachmeeting 2018-04-17

Present:

Simon Emanuel Schmid Benjamin Zumbrunn Sarah Hauser (FHNW)

Markus Knecht (FHNW)

### Agenda

Present and discuss documentation outline Discuss and approve prototype specification Discuss and approve project goals

Discuss more detailed project plan (Github) Possible collaboration for the [Melon Hackathon](http://hackathon.melonport.com/) Citations?

### Documentation outline

The content of the report is good, but the structure is not as intended. Write as if the project is finished, e.g. do not include something like a step by step report. Place planning, iterations, milestones, variants, prototype etc. on the end of the document.

Research -> State of the art. Or choose a different term, Ms. Hauser will supply one Delete conclusion & approach

New chapter Analysis: summary of what we learned from the "research", Evaluate chapter: in appendix

Design + Production -> new Chapter Solution We propose a new outline via mail.

### Protoype specification

The MIP is a good idea, but we should not focus on creating the process to define such MIPs (do not waste time...). Document the concept of *Contract Standard Proposal* in the chapter *state of the art*. Standard Report Mockup Draft is a good idea. Interchangable Fund Data Format.

Think about what we sign --> only the raw data (Melon Report)? An audit about the data?

Melon report: The JSON representing the factsheet must be constructed to also be extensible for [KIID](#_bookmark75), PRIIPS etc. It would be really nice to have a JSON that is also extensible for the "new world".

Knecht: We should not care about security considerations about signing (e.g. do I sign the data I have just seen?), but care about signing on protocol level.

Knecht: it is important that we work with ONE specific version of the protocol, e.g. the state of the competition in February(?).

Important: Define list of use cases, actors (regulators etc.).

### Project goals

Before specifying project goals (deliverables), define what a user can do with our project. New milestone for 3.5.2018: Use cases are defined and agreed on.

Define the goals from the *user perspective*, **not** from the technical perspective.

Show the current project goals in the meeting with Melonport as *ideas*. Knecht: Specify the *what*, not the *how*.

### Detailed project plan (GitHub)

Do not include the milestones in the "Kanban" board.

### Melon Hackathon

Simon will write Mr. Knecht about the hackathon.

### Citations

Use the IEEE format (with numbers), maybe there is a plugin. Ms. Hauser sent a mail with a link. Maybe use LaTeX when citation does not work at all.

### Expert

The expert for the project will be Konrad Durrer (CSS). He organizes a midterm presentation in May.

## Coachmeeting 2018-03-27

Present:

Simon Emanuel Schmid Benjamin Zumbrunn Sarah Hauser (FHNW) Markus Knecht (FHNW)

### Agenda

Present updated project plan (3 iterations) Discuss current state:

State of research State of [solidity](#_bookmark81)

Fix date for thesis defense

### Bachelor thesis defence

Accoring to Ms. Hauser, our defence can be held in the first week of the presentations: September 3rd to 7th.

### Project plan

There are some mistakes on the project plan:

prototype date is wrong -> 19.4.

first iteration -> should be second iteration

For 19.4.: attach to milestone: fix project plan with more details

For 17.4.: send coaches a draft of the documentation (list of contents is fixed)

After the first draft of the product, the first draft of documentation will be reviewed by coaches (5.7.). For 5.7.: everything about the final product must be decided

requirements fixed

all documents available all data available

all prototypes ready for extension

### State of research

Simon went through the research chapter with the coaches and discussed what he is working on right now.

Decide soon: which market will the product be for? (europe/swiss) Hauser: do not solve the problem of "different markets"

Define exact terms for the "technical" world and the "customer" world that will be used througout the project.

For the risk profile: Suggestion: if we cannot calculate the "monte carlo" risk, we should show that it is possible with the data that we have.

### State of solidity

Benjamin showed the draft of the auditing contract prototype and discussed the result with the coaches. Notes from Mr. Knecht:

bytes32 signature to bytes32 rs components doesn't make sense (information loss) Assembly code is critical: maybe deliver rsv of the signature directly

Check if delivering a signature is really important. (The 'sender' of the audit already delivers a signature with their address)

Define prototype specification with use cases.

## Kick-off meeting 2018-03-07

Present:

Simon Emanuel Schmid Benjamin Zumbrunn Mona El Isa (Melonport) Reto Trinkler (Melonport) Sarah Hauser (FHNW) Markus Knecht (FHNW)

### FHNW

Ms. Hauser is away for the whole month of may Always send protocols from meetings to both coaches

Always send decisions about the project (with melonport) to both coaches

### Thesis release

A poster and an interactive demo is required

Two printed versions of the thesis must be provided for the FHNW Send in a draft of the thesis as soon as possible for review

Plan three iterations of work/evaluate

Define milestones and give more detail on planned work

### Project

We might be able to assemble the benchmark data from coinmarketcap or similar platforms. Our Gitbook solution for project reporting is agreed by the coaches.

Possible verification of the generated reports:

History of all concerning trades

Hyperlinks which point directly to the blockchain on IPFS (see IPFS Mesh)

### Dates

We will send a doodle with tuesday-dates (3pm-6pm) for coach reviews at Campus Brugg/Windisch

### Progress Meetings in Zug

We already fixed the following dates for progress meetings with all stakeholders. They will happen in the Melonport Office in Zug.

May 3, 14:00: Presentation of the prototype

July 5, 14:00: Discuss Design & Architecture before final sprint August 6, 14:00: Last meeting before submission

# References

|  |  |
| --- | --- |
| 1 | [SFAMA, Vorlage für das KIID für Effektenfonds sowie für übrige Fonds für traditionelle Anlagen in der Form von Publikumsfonds (gemäss SFA-KIID-Richtlinien), *Swiss Funds & Asset Management*](https://www.sfama.ch/de/selbstregulierung-musterdokumente/fondsverwaltung/verfuegbare-dokumente/sfama_d_vorlage-fuer-das-kiid-fuer-schweizerische-fonds.docx) *Association*, 2012. |
| 2 | [SFAMA, Richtlinien zu den „Wesentlichen Informationen für die Anlegerinnen und Anleger“ für Effektenfonds sowie für übrige Fonds für traditionelle Anlagen in der Form von Publikumsfonds](https://www.sfama.ch/de/selbstregulierung-musterdokumente/fondsverwaltung/verfuegbare-dokumente/sfama-rl-kiid_d_richtlinien-info-anleger_120120.pdf) , *Swiss Funds & Asset Management Association*, 2012. |
| 3 | Ethereum Foundation, [web3.eth.sign](https://github.com/ethereum/wiki/wiki/JavaScript-API#web3ethsign), 2018. |
| 4 | [Ethereum Foundation, Units and Globally Available Variables - Mathematical and Cryptographic Functions, 2018.](http://solidity.readthedocs.io/en/latest/units-and-global-variables.html#mathematical-and-cryptographic-functions) |
| 5 | Melonport, [What is Melon, the 2018 Edition](http://www.medium.com/), 2018. |
| 6 | Ethereum Foundation, [Ethereum JSON RPC](https://github.com/ethereum/wiki/wiki/JSON-RPC), 2018. |

## ABI

Application Binary Interface <https://solidity.readthedocs.io/en/v0.4.24/abi-spec.html>

## ACID

Atomicity, Consistency, Isolation, Durability <https://en.wikipedia.org/wiki/ACID>

## AUM

Assets under management. A number indicating the total value of all assets in a fund denominated in the funds quote asset.

## CAP

Consistency, Availability, Partition tolerance <https://en.wikipedia.org/wiki/CAP_theorem>

## CISA

([FINMA](#_bookmark72))

Collective Investment Schemes Act / Kollektivanlagengesetz, KAG <https://www.admin.ch/opc/en/classified-compilation/20052154/index.html>

## CISO

Collective Investment Schemes Ordinance / Kollektivanlagenverordnung, KKV <https://www.admin.ch/opc/en/classified-compilation/20062920/index.html>

## CISO-FINMA

Ordinance of the Swiss Financial Market Supervisory Authority on Collective Investment Schemes / Kollektivanlagenverordnung-[FINMA](#_bookmark72), KKV-[FINMA](#_bookmark72)

<https://www.admin.ch/opc/en/classified-compilation/20140344/index.html>

## DAI

[Dai](#_bookmark66) is a cryptocurrency that is price stabilized against the value of the U.S. Dollar. [Dai](#_bookmark66) is created by the [Dai](#_bookmark66) Stablecoin System, a decentralized platform that runs on the Ethereum blockchain.

<https://makerdao.com/>

## EFAMA

European Fund and Asset Management Associaton <http://www.efama.org/SitePages/Home.aspx>

## ERC

Ethereum Request for Comments. A system similar to [IETF's RFC](https://www.ietf.org/rfc.html) where developers can propose standards. The most known [ERC](#_bookmark68) is [ERC-20](https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md) from Fabian Vogelsteller and Vitalik Buterin.

## ERM

Entity relationship model. A technical tool to model and visualize entities and their relations.

## EMT

European [MiFID II](#_bookmark76) Template

## EPT

European Working Group [PRIIP](#_bookmark78) Template

## ESMA

European Securities and Market Authority [https://www.esma.europa.eu](https://www.esma.europa.eu/)

## ETF

Exchange-traded fund. Often used synonymous for a fund that automatically tracks an index. <https://en.wikipedia.org/wiki/Exchange-traded_fund>

## FCA

Financial Conduct Authority UK <https://www.fca.org.uk/>

## FINMA

Swiss Financial Market Supervisory Authority / Eidgenössische Finanzmarktaufsicht [https://www.finma.ch](https://www.finma.ch/)

<https://www.finma.ch/en/authorisation/institutions-and-products-subject-to-the-collective-investment-schemes-act/>

## Inception

In asset management [inception](#_bookmark73) is the date of the beginning of a fund.

## ISIN

International Securities Identification Numbers.

ISINs uniquely identify a security -- its structure is defined in ISO 6166. Securities for which ISINs are issued include bonds, commercial paper, equities and warrants. The [ISIN](#_bookmark74) code is a 12-character alpha-numerical code. Think of it as a serial number that does not contain information characterizing financial instruments but rather serves to uniformly identify a security for trading and settlement purposes.

<https://www.isin.org/>

## ISO 4217

Currency codes, usually 3 characters whereas the first two are indicating the country and the 3rd the currency name. Example: CHF, USD, ...

Special currencies start with an X. Therefore, XBT would be the official code for Bitcoin, this isn't yet standardised.

## KIID

Key Investor Information Document. A summarized 1-2 pages document that contains most relevant documentation for retail investors.

## MiFID II

EU Markets in Financial Instruments Directive. MiFID I originally from 2004 it's successor [MiFID II](#_bookmark76) took effect on January 2018.

<https://en.wikipedia.org/wiki/Markets_in_Financial_Instruments_Directive_2004>

## MOP

Multiple Investment Options

## Participation

[Participation](#_bookmark77) is the act of investing or redeeming shares of a fund.

## PRIIP

Packaged Retail and Insurance-based Investment Products

A [PRIIP](#_bookmark78) is defined as: an investment where, regardless of its legal form, the amount repayable to the retail investor is subject to fluctuations because of exposure to reference values or to the performance of one or more assets that are not directly purchased by the retail investor; or an insurance-based investment product which offers a maturity or surrender value that is wholly or partially exposed, directly or indirectly, to market fluctuations.

The aim of the [PRIIPs Regulation](http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX%3A32014R1286&amp;from=EN) is to encourage efficient EU markets by helping investors to better understand and compare the key features, risk, rewards and costs of different PRIIPs, through access to a short and consumer-friendly Key Information Document (KID). How information in the KID should be calculated and presented is set out in the [PRIIPs Regulatory Technical Standards (RTSs)](http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1516899203728&amp;uri=CELEX%3A02017R0653-20170412).

Source: <https://www.fca.org.uk/firms/priips-disclosure-key-information-documents>

## Prospectus

A [prospectus](#_bookmark79), in finance, is a disclosure document that describes a financial security for potential buyers. <https://en.wikipedia.org/wiki/Prospectus_(finance>)

## Sketch

Digital Mac OS X software for UI/UX Designerss

## Solidity

Specialized language to develop smart contracts

## SRRI

The synthetic risk and reward indicator ([SRRI](#_bookmark82)) is used to classify investment funds into one of three different risk categories (low risk, medium risk, high risk). It is calculated on the basis of Austrian and European regulatory requirements. This indicator forms an integral part of the Key Investor Information Document ([KIID](#_bookmark75)) and gives the historical volatility of the fund unit price on a scale from 1 to 7.

|  |  |  |
| --- | --- | --- |
| [**SRRI**](#_bookmark82) | **Risk category** | **Volatility intervals** |
| 1 | Low risk | 0% to <0.5% |
| 2 |  | ≥0.5% to <2.0% |
| 3 | Medium risk | ≥2.0% to <5.0% |
| 4 |  | ≥5.0% to <10.0% |
| 5 |  | ≥10.0% to <15.0% |
| 6 |  | ≥15.0% to <25.0% |
| 7 | High risk | ≥25.0% |

<http://fundglossary.erste-am.com/srri/>

[CESR’s guidelines on the methodology for the calculation of the synthetic risk and reward indicator in the Key Investor Information Document](https://www.esma.europa.eu/sites/default/files/library/2015/11/10_673.pdf)

## UCITS

Undertakings For The Collective Investment Of Transferable Securities

<https://www.investopedia.com/terms/u/ucits.asp> <https://en.wikipedia.org/wiki/Undertakings_for_Collective_Investment_in_Transferable_Securities_Directive_2009>

## VaR

Value at Risk

# Original project submission

**Declaration of Academic Integrity**

Hereby, we declare that we have composed the presented thesis independently on our own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such.

This thesis has neither been previously submitted to another authority nor has it been published yet. Place, Date

Signatures

# List of figures

**List of tables**