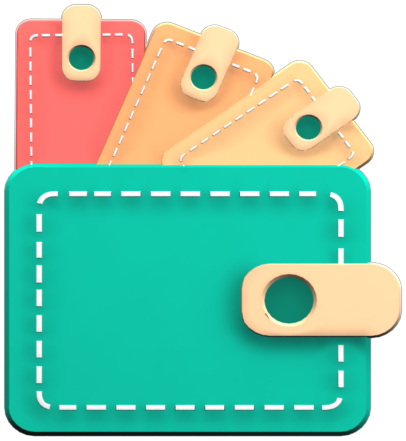
Fermat

expanding the cryptoworld

Gustavo H. Corradi <book@fermat.org>

v.0.0 (draft), november, 2015.

Fermat Book



Fermat

Introduction

What is Fermat?

Fermat is a **trust-less financial application framework**, which contains a Multi-Digital-Wallet system that handles both *crypto currencies* and *digital assets* . It allows non-developers to create their own digital wallets or trust-less financial applications reusing or combining other wallets or wallet components built into the system as plugins. It also holds a P2P network for transporting meta-data and inter-connecting clients one to the other.
Fermat is a set of services facing the end user and based on the existing blockchain protocols.
In order to bridge the existing gap between the underlying crypto technology and the end user, Fermat offers a first layer of *reference wallets* that are developed to address different use cases, one per wallet.
A second layer of *niche wallets* combine reference wallets functionalities and add their own touch.
This both enables a third layer of ***branded wallets*** that can be generated just by changing the look and feel and navigation structure of niche wallets without the need of a

developer.
The goal is to allow any company big or small, brand or institution to have their own digital wallet with **any** subset of the whole functionality built for the entire user community.
Fermat's architecture includes the possibility of interconnecting the trust-less financial applications developed within its framework to the existent legacy financial system.

Fermat Community

Fermat is an Open Source project and there is an active community of developers, designers and academics contributing to it. The project's maintainer community is bitDubai, a joint venture of some of them.

The project started on August 2014 and the early first version of its core was written from January to May 2015 by a handful of people. From there many more developers joined in. It is **code base portable** because mostly of it is written in **JAVA**.

Today Fermat is a vibrant community spanning several countries and disciplines.

The Vision

The Fermat Community believe that the next decade will witness **crypto currencies** going mainstream and governments issuing digital currency as well. They expect **paper money** is going to *disappear* within the next 5 to 10 years. Thus a strong consumer demand for **digital wallets** will inevitably emerge. In order to meet this demand thousands of different wallets should be available, different not only in terms of the level of abstraction they apply (some of them showing crypto currencies while others digital fiat over crypto or even digital assets) but also with respect to their appearance and how they feel to the user, as well as differences in their functionalities.

The Fermat Community is contributing to that future with a System to build and run any kind of digital wallets easily by any developer or even non-developers.

Fermat's Purpose

- Empower every person in the world with the appropriate tools to save and move their money in a digital format in an efficient, secure an private way.

- Give back the sovereignty over each one's own resources, respecting the freedom of choice about which currency to use and how to administrate it, regardless of the place they live or where they were born.

Fermat's Mission

- To get cryptocurrencies mass adopted as soon as possible.
- To build the best possible integrated system able to handle the finances of not only all the human world population but also of companies and machines in the context of the IoT.

The challenge

The design of Fermat addressed the answers to the following questions:

- How can the perception of complexity on crypto currencies by ordinary people be isolated?
- How can the collective intelligence of the developer community be used?

- How can the entire current infrastructure already in place be reused?
- How can legacy financial institutions be integrated in a way they feel comfortable?
- How can merchants, retailers and brands of all sizes be integrated in a clever way?
- How can non-developers be allowed to create digital wallets without programming anything at all?

The answers to each of these questions are addressed within Fermat.

Fermat's Principles

The community developing the Fermat System agreed on the following founding principles:

1. Fermat must not allow censorship.
2. Fermat must not allow spying on their user base.
3. Fermat must be secure and resistant to all kind of attacks.
4. Fermat must never loose a user's funds or assets.

5. Fermat must be useful to each segment of the world population.
6. Fermat must be extensible and open to innovation with a master plan approach.
7. Fermat must be open to any developer to participate.
8. Fermat must compensate each developer by their contribution.
9. Fermat's user base must be a shared asset.
10. Fermat must be inclusive with crypto currency industry members.
11. Fermat must be inclusive with the legacy financial industry members.
12. Fermat must be un-banked-people friendly.
13. Fermat must be OS agnostic.
14. Fermat must learn from its user base.
15. Fermat must be the financially most efficient way to hold, move or spend the end user's money.
16. Fermat must facilitate the regional distribution and access to crypto currencies.

Fermat Book

Learning about Fermat is very easy. It's just a matter of continuing reading this book that will guide you step by step all the way until you reach a complete understanding of this amazing technology.

Continue Reading ...

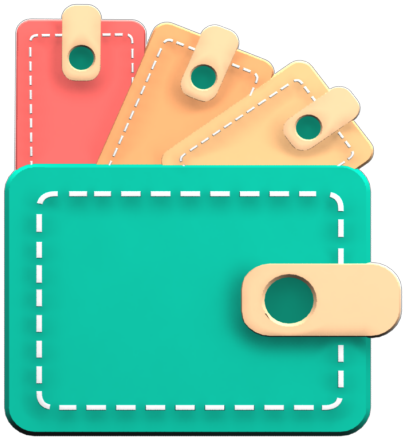
Next Chapter

Appendix: Fermat Principles in Detail

Appendix: Fermat History

Fermat Network Visualization

If you are more confortable with an overview of Fermat rather than reading, please refer to the official site http://fermat.org



Fermat

Chapter 1: Fermat Master Plan

As a Master Plan, Fermat has it’s own vision of how a collaborative project should be carried out. The main concept is to define a large master plan far into the future, and start building it phase by phase adjusting the target when needed. It is designed in this way to absorb the efforts of thousands of contributors over the years into a unified architectural vision.

After the pioneer step of Nakamoto who gave rise to **Bitcoin** and its revolutionary blockchain technology, in the last years, there has been a lot of fin tech investment money given to a myriad of startup initiating in this industry. This fostered the next evolutionary step, **"cryptocurrency 2.0"** but it has still not reached the level of adoption expected. There seems to exist an unbalance between the considerable investment of funds and effort needed to master this technology and the results they are bringing up. What is more, isolated projects trying to build something useful starting form zero point have it difficult to obtain the

necessary support and the persistent endurance needed to succeed…
Precisely acknowledging this weak points in the industry, and choosing **collaboration** within a **master plan** as starting point, it is how the Fermat Community believes to succeed in solving the challenges set in their mission and purpose by building a complete new framework that solves the problem in an efficient, inclusive and attractive way.

Phase I: Foundations

The goal is to have a framework for developing financial applications in a trust-less environment. What are the services that had to be granted at the lower levels to enable this?

The basic infrastructure

> the *Fermat Core Platform* is the part of the system which initializes the entire set of components, starting every plug-in and putting them to run. This creates the interconnection that let all the services to be offered as being integral part of one unified system.

The basic services

connection to the different Operating Systems that will run Fermat on a chosen device.

- interconnectivity to cryptonetworks and blockchain services providers to enable the **transport of value**.
- P2P communication of the devices running Fermat to enable the **transport of information (meta-data)**

The plug-ins environment

Fermat Plug-ins Platform deals with this task allowing systems users to be identified, low level plugins to be recognized as part of the system, communications services to link plug-ins living in different devices and in a general way,..an so on. In general terms, all these underlying construct allow various *network services* to run and start building the set of functionalities that will be taken by the applications built on top.

NOTE

At this stage of development, Fermat already offers a wide scope of REUSABLE components for third parts to build on top, encouraging further development of the plan and enrichnening it with every single addition.

Phase II: First constructs

The foundations have been laid out, but to reach the end user oriented applications, it is necessary to create several functionalities. In order to make a visible structure available for further developments and also to provide certain focus to related functionalities, these are wrapped in substructures of Fermat system called **platforms** and **superlayers**. Each of them groups a subset of functionalities that offer the following services :

- a decentralized Wallet Production Line
- a Crypto Currency platform to manage the transference of value trough crypto transactions
- a Shoping platform to vinculate shop owners with brands and their customers
- a Digital Asset platform to allow issuing a vast type of digital coupons exchangeable for services and products
- a Marketing platform based on Digital Assets
- a Crypto Broker platform to easily administrate the business of exchanging crypto and fiat currency
- a Crypto Distribution Network to create local

commercial environments fostered by crypto money

among others...

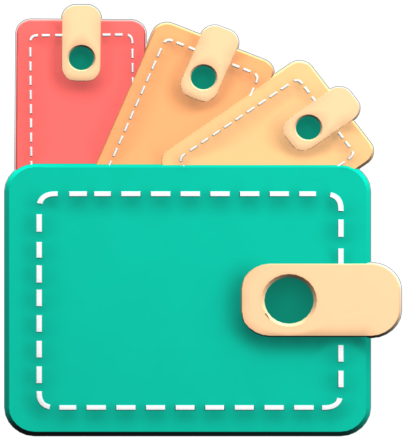
Phase III and beyond:

Have you grasped the *potential* of Fermat ? Can you imagine the next phases of development ? Are you interested in getting to know Phase I and II deeper?
Follow next chapters to get a more detailed description of each one.

Continue Reading ...

Next Chapter

Previous Chapter



Fermat

Chapter 2: Fermat as a Platform Stack Framework

Fermat is, technically speaking, an Open Source P2P (Peer-to-Peer) Platform Stack Framework built on a Plug-ins Architecture running on end-user's devices.

Open Source
: the code can be reviewed and audited by anyone.

P2P network and decentralized
: censorship resistant and difficult to be attacked.

Platform Stack
: functionalities are wrapped in categories easy to grasp

Plug-in Architecture
: reuse of components and open to the participation of any developer.

Ir runs on end users' devices

users control on their money without the need to trust a third party.

What is Fermat made of ?

- **libraries**: structural components of the core of the system (low level layers)
- **desktops**: applications that run on an specific Operating System and present the GUI (graphic user interface)
- **plug-ins**: components which encapsulates a very specific sets of functions, consume services from other components and offer a public interface for the services they provide to other components. They are open to be developed by the general developers community.
- **add-ons**: plug-ins in the low level layers that provide the most sensitive functionalities to the system, and therefore are developed by trusted developers.

Fermat’s architecture

Fermat as a system is built on several overlapping architectural paradigms:

1. At a higher level, it is a **Peer to Peer system with asymmetric nodes**, meaning that each node even having all the code base, it specializes itself according to the profile of the end user or the way it was configured.
2. Inside each node Fermat features a **Multi-OS architecture** meaning that the lowest level OS-dependent components are wrapped in a way that can be easily replaced when running on a different OS without affecting the rest of the components consuming services from them. The up-most components, the ones facing the end user (GUI) are also OS-dependent. Everything in between is **not**.
3. At the same time it features a plugins architecture inserted into a **multi-layered structure**. These layers are occasionally grouped into **Superlayers**.
4. These **plug-ins** are subdivided into a hierarchy of **Platforms**. Those **Platforms** shares a common set of layers, and each one adds **Actors** and **Products** to the overall functionality of the Fermat system. Inside each Plug-in, you might find a specialized structure and in many cases a database or files belonging to that plug-in.
5. At a system level, Fermat uses specialized crawlers to collect or sometimes inject information into nodes in

order to recreate a system level consciousness prepared to resist attacks or other relevant issues.

p2p architecture

Fermat is a *distributed system*, it runs on end user's devices, so it needs to establish a connection between every actor and every device using it. This is done by creating a **P2P network** between them, based on the services provided by *Fermat P2P Network and Communication Superlayer*.

multi OS architecture

Fermat is **code-base portable**, meaning that the components are written in code that runs in different operating systems. So, through *Fermat Operating Systems API Superlayer*, the components that are specific to a given Operating System connect to the rest of Fermat that is OS independent.

multilayer architecture

Fermat needs to be adaptable, to provide highly component reutilization, and to offer a scalability that match the complexity of the services that are running on it. Therefore

the components are set in **layers** stacking one over the other. Each layer groups components that provide a similar functionality, eg. communications layer is designed to provide a way to connect one device to another building a "communications channel" using different available technologies (cloud servers, wifi, NFT, etc), and in performing this task, it let other components from other layers do their specific task consuming services of this communication layer in a transparent way (without any concern on how the communication is actually being established).
There are certain **Layers** that provide services at a system-wide scale (see **superlayers**), but other layers are defined within one **platform** and serve to provide its set of specific functionalities.

plug-ins architecture

Plug-ins have an outstanding feature: as long as their INTERFACE (i.e. the shape of the interconnection) is known and public, the plug-in can interact with others, consuming and offering services, INDEPENDENTLY of its inner structure and how it is built. As long as they conserve this INTERFACE, they can safely evolve to more efficient functioning WITHOUT altering a single service that has

been built upon it ! Inside Fermat, each plug-in is given certain specific responsibility within the whole, it lives in a certain layer, and it is allowed to consume services of components on *lower layers* and to provide services to components of *upper layers*. Plug-ins participate in high level processes and they are programmed in a way to live in an uncontrolled environment (end user devices) and to co-exist with untrusted third party plug-ins as well. Through this, Fermat is able to connect and reuse most of the infrastructure deployed by the industry.

platform architecture

A **platform** consists of a group of components living in different **layers** interconnected to offer a specific set of services for a discovered niche. For example, if we address to cryptocurrency users, we will need a *cryptocurrency wallet* for each cryptocurrency available running on *Fermat Crypto Currency Platform*. This wallet would operate on the selected cryptonetwork by means of a connection to it provided by the *Blockchain Platform*, and will interact with the user by means of an **desktop** living in the OS specific layers.

Following the Master Plan

Once we have described Fermat's components and the architecture, we will explore in the next chapters the platforms that implement the foundations (Phase I) of the master plan:

- **Fermat Core Platform**
- **Fermat Operating Systems Superlayer**
- **Fermat Blockchain Superlayer**
- **Fermat P2P Network and Communication Superlayer**
- **Fermat Plug-ins Platform**

For a cool visualization of Fermat and its constant growing number of platforms and superlayers visit http://fermat.org We will cover the platforms and superlayers from bottom to top and from left to right, as they are presented in the visualization.

So, let us enter the "building site" and explore the foundations.

Continue Reading

Next Chapter

Previous Chapter


Fermat COR
Core & API

Chapter 3: Fermat Core Platform

This is the **core platform** of Fermat Multi-platform System. It contains the deepest levels of layers in the layers stack. Its components encapsulate the basics definitions and functionalities that let Fermat operate as an *integrated system*.
The platform first defines a *plug-in identity* for each plug-in (this is done only ONCE for the whole system), which let the plug-in be **recognized** as member of Fermat and then access to certain data specifically owned by it (like accessing to the file system or data base systems, etc). Then it put them to run (see [system initialization]) and builds the pattern of relationships between the isolated plug-ins, thus deploying all the system wide infrastructure.
Fermat Core Platform is written in JAVA as the most of Fermat, and that there is a JAVA implementation for each known operating system, that makes Fermat **code base portable**.
After initialization, this platform monitors the way the system is behaving as a whole.

Platform components

fermat core

Initializes all system-wide plug-ins and start a specific **core component** for each platform/superlayer running on Fermat, which in turn initializes the platform/superlayer's specific set of plugins.

android core

Initializes android's components that connect Fermat to the android user's environment.

platform specific core

There's a **core component** for each one of the platform existent in Fermat, which creates the *plug-in identity* and initializes all platform/superlayer's specific plug-ins:

===

Continue Reading ...

Next Chapter

Previous Chapter


Fermat OSA
Operating System API

Chapter 4: Fermat Operating Systems Superlayer

NOTE	We call in Fermat **Superlayer** to a group of layers whose components offer services to be consumed system-wide, i.e. offering an infrastructure that hold the different **platforms** running over Fermat. (it is like a "common ground" where different "towers" (the platforms) are built, sharing basic level services)

This superlayer allows the connectivity of OS dependent components to the rest of Fermat multi-platform that is *OS independent*. The components offer the necessary functions to interact with the corresponding OS, like storing and retrieving information, managing files, managing user log-on, checking the device status, location, etc
Specific layers will be added as long as Fermat’s releases

include new OS.
This low level layer allows to build on top in successive layers the components necessary to implement the **Desktop Applications** offered to Fermat end users.

Android layer

In this layer, we find the different components to allow Fermat function on a device running Android. They are written in *android language*.

File System

it allows creation of files in a reserved memory area for each plug-in (it functions as a "private folder")

Database System

it allows creation of a database only readable for the plug-in who owns it.

Logger

access to the OS activity log.

Device Location

access to the GPS coordinates to determine device location.

Device Connectivity

: access to peripherals connecting to the device through USB Port / Bluetooh Port / etc..

Device Power

Device Contacts

Device Hardware

: information about the hardware of the device running the platform

Multi OS layer

In this layer there are components that run in more than one OS at the time (Linux / MacOs / Windows).
The components are similar to the other *OS layers*.

I-OS layer

(open for further development) ...

Continue Reading ...

Next Chapter

Previous Chapter


Fermat BCH
Blockchains

Chapter 5: Fermat BlockChain Superlayer

This superlayer provides the entire Fermat multi-platform system the necessary functionalities to manage **blockchain technology**: connecting to the *crypto networks*, reading the *transactions*, "storing" the value related to Fermat end user's transactions, etc.

Crypto Networks layer

This layer builds the interface between Fermat and each *cryptonetwork* available.
There is one component for each cryptocurrency: **Bitcoin**, **Litecoin**, **Ripple**, **Ethereum**, etc
The main *responsibility* for each component is to handle every outgoing/incoming request to/from the corresponding cryptonetwork. More specifically, the component connects to a **full node** of the corresponding cryptonetwork and explores the transactions included in the blockchain to find where a **Fermat public cryptoaddress** is mentioned. When a match is found, the component races en event

broadcasted in Fermat environment, which in turn triggers some action in another component which *responsibility* is linked to the event. The addresses to match are those listed in the *Cryptoaddress Book* component (see *Crypto components layer*).

Crypto Vaults layer

As well as in a traditional **funds management** of a Bank, the entire money that a Banks possess is store in one *main vault*, and the *user's account managing system* acknowledges how much money corresponds to every one, in Fermat, all transactions corresponding to Fermat's users are held in a **vault component**, corresponding to the currency associated to the transaction. Then is task of every Fermat **wallet** to manage the amount of cryptocurrency owned by each **actor**, in the way wallets do: they track every **unspent transaction output** in the *blockchain public ledger* and they display the information (transaction history, transaction report, balance, etc) according to their built-in functionalities.

NOTE

It might be useful to recall in this section a basic concept in blockchain technology about Hierarchical Deterministic Wallets. These Wallets are created in a way that they generate a **master private key** and taking this key as a **parent branch** of a derivation tree, a complete set of **children** keys might be derived from it in a hierarchical deterministic manner. This enables the creation of a very large number of children private keys to sign transactions and children public key to obtain valid cryptoaddress from them -with the usual procedure of applying a *hash function*- which allow the derived wallets created in the system to properly operate with them *without* revealing the **master key**.

So, each **vault component** uses its *master key* in the way of a HD wallet, to issue derived children keys which supply the necessary keys to each platform within Fermat to perform valid cryptocurrency transactions with complete safety and minimizing the data size to be shared and transmitted along the system.

Bitcoin Vault

it stores the entire system's children keys that let all Fermat Bitcoin Wallets **unlock** the funds owned by their users, so it can be thought that this vault "stores" the available *Bitcoin* funds of the entire system. (**value storage**).

Assets over Bitcoin

it "stores" (in the sense above mentioned) each little amount of_Bitcoin_ used to validate a **Digital Asset** (called assets *proof of existence*). It is more a **meta-data storage** of what is represented by the **asset** than a a storage of **value**.

Bitcoin Watch Only Vault

it stores "receive only" type of Bitcoin transactions, what means that the transaction is showed in the ledger, but there is no *private key* available to claim the Bitcoins transfered. This type of **addresses** are used in the *Digital Asset Platform* to allow **redeem points** wallet to work, showing they have received an Asset to be redeem by the **Assets Issuer**.

Crypto Components layer

Crypto Address Book

it keeps a list of every cryptoaddress requested by any component of Fermat, recording who originated the request, from which platform, and any other necessary information to identify the address when consumed from any plug-in. It is the *authority* to recognize every address belonging to Fermat from the rest of addresses available in the blockchain ledger.

Crypto Router layer

This layer really acts as a **router** of incoming/outgoing transactions referring to Fermat's users, listening to the specific events to be aware of the transaction and "delivering" them to the appropriate plug-in to handle it.

Incoming Crypto

it receives a call from the *Crypto Network Layer* when a Fermat address receives a transfer of cryptocurrency (what is identified by consulting the *Crypto Address Book*), and it races an event to pass the responsibility to the corresponding plug-in to handle this transaction.

Outgoing Crypto

it delivers the transaction to the corresponding **cryptonetwork** and tracks the confidence level to inform the wallet the status of the transaction.

Continue Reading

Next Chapter

Previous Chapter


Fermat P2P
Peer to Peer Network

Chapter 6: Fermat Network and Communication Superlayer

This superlayer provides connectivity between **devices** by establishing a *communication channel* between a plug-in running on a **local device** which requires to consume certain service from its *counterpart* (i.e. the plug-in with the same task) running on a **remote device**.
This superlayer takes the entire responsibility of maintaining the communication channel "alive" as long as it is required, providing this **service** as a whole, i.e. offering this channel to the external layers and plug-ins in a **transparent** way for them, regardless of the **communication technology** that the layer uses to build this channel ***dynamically***.
The upper layers should not be aware of the technology in use (wi-fi, NFC, p2p, bluetooth, etc.) and problems related to

a connection should be solved by the communication layer without disturbing the clients.
This ensures a safe way to offer **network services** built on top of this layer, not only for Fermat developers but also to third party.

Communication Layer

This is the only layer in this superlayer which provides the communication services as a whole, i.e. components that need to be communicated interact with the **layer interface** and not directly to any plug-in inside. Each one of them implements the communication protocols and interfaces for one of the different **communication technologies** available.

Cloud Client

> This plug-in manages the requirement of a **client** plug-in to establish a connection to a *counterpart* plug-in running on another device. It transmit the call to the *cloud server* which "switches on" the communication channel to the destination party.

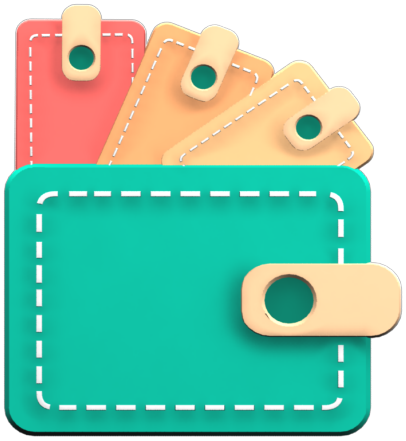
Cloud Server

For certain basic communications demand, a cloud server is a reasonable good option. At initialization of Fermat, the cloud server manager receives the initial "log in" signal of each service that is starting, and keeps record of their "on-line"status. Then, when a *local* **network service** needs to contact its *counterpart* , the cloud server has the necessary information to establish the **communication channel** requested.

Continue Reading ...

Next Chapter

Previous Chapter



Fermat

Chapter 7: Fermat layers explanation

The ***multilayer architecture*** reveals useful when grouping the plugins according to their common functionalities. This gives , on one hand, a certain order and structure to the system and lets identify common challenges to solve when writing code. On the other hand, the conceptual strength of using *layers* stimulate the creation of future plugins that are revealed as necessary when decomposing a big task in smaller steps to solve and implement. What's more, plugins that are originally conceived to serve a certain function, afterwards appear to be reusable for new developments beyond their initial purpose.

We have already explored in the previous sections the main role of the **Core Platform** to deploy the whole system structure, and the first available services provided by the system-wide ***superlayers*** .

Now, let us explore the next ***layers*** following on top, which are almost present in every ***Fermat platform*** which follows.

Layers are built with a considerable atomicity in its

components (i.e. each component deals with a very specific and constrained set of events) what really renders the whole Fermat system with a strong, structured and solid foundation and at the same time a large flexibility and adaptability for further development and evolution.

1.1. *Platform Services layer*

The modules in this layer handle the very basic tasks that allow plugins interact safely in and uncontrolled environment, taking care of a careful transmission of responsibility from one component to another and specially the management of exceptions and track of all system's **events**.

1.2. *Network Service layer*

Each time that a certain system function must perform any operation **outside** the device where Fermat is running, a *network service* is required to follow the execution in other devices until is completed.
Network services' modules need to have a **communication channel** established and maintained between the parties involved, what is done by the **Fermat network**

Each **network service module** running on a *local device* establishes communication EXCLUSIVELY to its *counterpart* running in the *remote device*.
The module handles a table of request and responses to communicate with the counterpart and so perform the required task.

1.3. *Actor Network Service layer*

This layer provides specific **network services** to interconnect the **actors identity** of an actor of a specific Fermat platform to the **user identity** on which it depends.

1.4. *Identity layer*

Plug-ins in this layer manage the necessary information about the end user's **identity** within Fermat. Every **identity** in Fermat is a pair of private/public key that acts like a credential to access to certain functionalities for specific to the * identity's owner For example, it allows the **end user** to have access to his own **wallets**.

1.5. *World layer*

In this layer the components consult certain specific information from the world "outside"of Fermat, usually checking indexes on web-pages, or gathering any kind of information offered in the web, to make this available to Fermat plugins to consume the collected data.

1.6. *Wallet layer*

The components in this layer do all the **accountancy operations** related to the wallets.

1.7. *Transaction layers*

Transaction

> a transaction is basically any operation that includes **money**.

A transaction is dependent on the *atomicity* of the changes of status in the operations involved, so each *transaction plug-in* should acknowledge whether the operation was successfully ACCEPTED or at the contrary, it was REFUSED.

Each plug-in deals with a certain *transaction type* defined by different criteria, mostly of the time, according to the **actors** involved and the **direction** (incoming / outgoing).
In Fermat the **transport of value** goes on a parallel path to the **transport of information (meta-data)** of the transaction, so both transports are managed by the *transaction plugins*.

Digital Money Transaction layer

This type of transaction layer deals with **digital money transaction**

Cash Money Transaction layer

This type of transaction layer deals with **cash money transaction**

Money Transaction layer

This type of transaction layer deals with **digital money transaction**

1.8. *Business Transaction layers*

More complex plugins that deal with multiple transactions that occur at the same time to reflect a desired **business transaction**

1.9. *Request layer*

Plugins in this layer gather the different *requests* than have been triggered in the system and deliver them to the appropriated plug-in to manage and fulfill the request.

1.10. *Middleware layer*

Every plug-in that cover any other functionality required and that is not included in the other layers.

1.11. *Actor layer*

This is the layer where the business logic of a specific **actor** is defined. **Actor identities** are derived in a hierarchical way from the end user's **root identity**, and let him create and control how many identities he wants to play different

roles within Fermat’s platforms.

1.12. *Agent layer*

High level plugins (near to the **end user**) that by means of a certain kind of *Artificial Intelligence (AI)* can take decisions on the **user’s** behalf, to simplify the management of wallets with high abstraction levels. For example, when the user wants to make a payment, an agent could choose where to withdraw the fund from to be more profitable financially to the user.

1.13. *Modules layers*

Modules plugins in this layers act as a "general manager" of all the functionalities that a **desktop** needs to consume from Fermat, delivering to it what is required in a transparent way (without the desktop program actually managing anything inside Fermat, but querying the *desktop module* to get what it needs).

Desktop Module layer

Desktop

is an sub-application (sub-program) that concentrate functionalities provided by several other sub-applications to present them in a *graphic user interface (GUI)* to interact with the user. (see ***Desktop layer***)

Sub App Module layer

Similar to the *Desktop Module layer* but applied to the *Sub Applications* that run **below** *desktops*.
These *sub app* do **NOT** manage money.

Wallet Module layer

Similar to the *Sub App Module layer* but applied to *Wallets*, i.e. "sub apps" that **DO** manage money.

NOTE	Until now, the *layers* presented are written in **JAVA**. The next layers are NOT **code portable**, i.e. they depend on the Operating Systems where they run, and therefore are written in each OS's specific language.

1.14. *Desktop layer*

In this layer live the part of the **desktop application** responsible of the GUI (Graphic User Interface) which has a **one-on-one** relationship to the component of the same name in the *Desktop Module Layer*

1.15. *Subapp layer*

In this layer live the part of the **sub app** responsible of the GUI (Graphic User Interface) which has a **one-on-one** relationship to the component of the same name in the *Sub App Module Layer*

1.16. *Reference Wallet layer*

In this layer live the part of the **wallet** responsible of the GUI (Graphic User Interface) and which manages the **wallets resources** (multimedia) and the **wallet navigation structure**. This also has a **one-on-one** relationship to the component of the same name in the *Wallet Module Layer*

Continue Reading ...

Next Chapter

Previous Chapter


Fermat PIP
Plug-ins Platform

Chapter 8: Fermat Plug-ins Platform

This platform encapsulates all the components that provide the basic services that build the back end infrastructure of Fermat framework.
An essential identity introduced in this platform is the **device user identity**, which is only valid within a specific device, and its public part is never revealed.
There is also an **actor identity** introduced: the **plug-in developer identity** what is essential to this platform because it puts together the basic functionalities that let Fermat function as a whole system, although it is made of distinct and separate components even created by different developers.

Plug-ins are arranged in the corresponding layers described in the previous chapter, starting from *Platform Services layer* at the bottom up to *Subapp layer* at the top.

Master Plan Phase II

Fermat foundations had been laid out by the five platforms and superlayers described in the previous chapters. Now let us enter the Phase II, where different platforms type are built: those to manage *crypto currencies* and those for *digital assets*.

Crypto Currencies platforms

1. **Fermat Wallet Production & Distribution Platform**
2. **Fermat Crypto Currency Platform**
3. **Fermat Crypto Commodity Money**
4. **Fermat Bank Notes Platform**
5. **Fermat Shop Platform**
6. **Fermat Cash Money Platform**
7. **Fermat Bank Money Platform**
8. **Fermat Crypto Brokers Platform**
9. **Fermat Crypto Distribution Network**

Digital Assets platforms

1. **Fermat Digital Assets Platform**

2. **Fermat Marketing Platform**

Accessory services

1. **Fermat Distributed Private Network**

Would you like to see the first tangible use of the supporting infrastructure of Fermat ? We invite you in next chapter to meet the *Wallet Production & Distribution Line*.

Continue Reading ...

Next Chapter

Previous Chapter


Fermat WPD
Wallet Production & Distribution

Chapter 9: Fermat Wallet Production & Distribution Line

Fermat builds the necessary services to run a wallet application at a user level collecting different functionalities available. Each functionality has its own work-flow, i.e the thread connecting the plug-ins from different layers to complete the task.
First group of functionalities are wrapped in the **reference wallets**, which are designed to fulfill only a specific objective.
Niche wallets are built combining a set of **reference wallets** to satisfy the specific needs of a certain type of **actor**.
* For example, Fermat *Bitcoin Wallet* is the niche wallet of the *Crypto Currency Platform*, designed for those actors that only require a wallet to manage their value in **Bitcoins** without further complexity. This is based on the *Crypto Wallet* as reference wallet, which in turns sets in motion the specific work-flows to fulfill its duties.

Inside a wallet, there are two distinct groups of the components:
* components oriented towards the user (what he sees and how he manages the wallet), called **front-end** components.
* components that perform each function of the wallet at operational level, called **back-end** components.

Fermat provides a *Wallet Manager application* to install available wallets in the user's device, collecting from the distributed network, the necessary front-end components (wallet's *navigation structure* and wallet's *resources*) and activating the dormant back-end components built-in the Fermat implementation.

Fermat Wallet Factory

Once we have at least one **Niche Wallet** (e.g.**Bitcoin Reference Wallet**), we can realize that in the **front-end** there are two distinct tasks taking place: on one hand, there exist a **navigation structure** (available menus, tabs, buttons, etc) that provides the means by which the user operates the wallet, an on the other hand, there exists a set of **resources** (multimedia files, images, etc) used for the visualization of each element of the wallet. At the **back-end**

, where the real action is occurring, we have a lot of components acting to provide the necessary functions, like:

- sending and receiving the cryptocurrency
- keeping track of the balance of the wallet
- managing the contacts (register, identification, etc)

If someone having the required programmer's skills wants to build a wallet in the traditional way in the industry (i.e. without using Fermat), he would necessary have to build every and each one of the components needed for the wallet to run writing the code from zero. However, *Fermat Wallet Factory* enables developers to take any of the available **niche wallets** to reutilizes its COMPLETE back-end (all the programming code stuff!), and concentrate his endeavor only to the **front-end** making his job a lot easier and allowing all his creativity to be channeled in a better and more attractive result for the end user. (not only for a nice look but also for improved and more efficient functionalities!).

Inside the Wallet Factory

The first step to create a wallet is to join Fermat an get an identification as **plug-in developer**. Then, by login into the

Wallet Factory application he choses a **reference niche wallet** to import, gives his wallet's project a new name and saves it as a new WFP-file (WALLET FACTORY PROJECT FILE).
Once in his **own project** he will re-define the **navigation structure** and then he will assign which **wallet resources** he will choose to give his wallet an unique appearance.
The last step is to run the project and test how it works until he is satisfied with the result.
To give birth the new wallet into the world, he will have to **publish** it using *Fermat Wallet Publisher*

Fermat Wallet Publisher

The Wallet Publisher takes the new wallet designed and held into a WFP-file and **publish** it in the *Wallet Store* and by this, it becomes available to the rest of the Fermat community to use it.
The publisher is also responsible of notifying the release of new updates for each version or upgrades to new versions of existing wallets.

Fermat Wallet Store

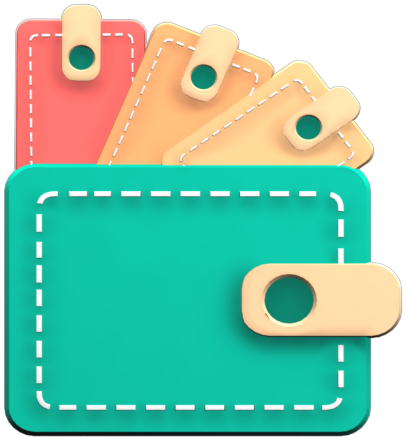
The Wallet Store component is inside Fermat a DAPP (Distributed Application), i.e. and undetermined amount of nodes collaborate between them to provide the functionality of the application inside the whole system.
The Wallet Store functions in a similar way of known app-stores, and it serves to show the user the wallet that may be attractive to him. It is a **distributed catalog** of wallets.
When one new wallet is chosen, the Wallet Store notifies the Wallet Manager to make it available to the user.
Internally, the Wallet Store keeps the identity of the wallet, of its developer, and of the resources used by it.

Let’s explore the different niches covered by Fermat in the platforms described in next chapters.

Continue Reading ...

Next Chapter

Previous Chapter



Fermat

Chapter 10: Fermat Money Management

Monetary transactions in Fermat are handled separating the **transport of value** (that follows an specific *work-flow*) from the accountancy of the money, and this all is separated from the transmission of the additional information of the transaction (the meta data).
Usually transactions start in specialized plug-ins of one of the transaction layers, an the work-flow's chain of actions reach to different levels, from consuming services of the lower layers such as the *Blockchain superlayer* to connect with the **crypto networks**, to the high level GUI components. The process interacts with Fermat's *vaults* in the *Crypto Currency Platform* to record the value, and contacts the **wallet** component to do the accountancy.
The rest of the information of the transacting not related to the value, is transmitted through the different *network services* through the Fermat Network which allow this **meta data** reach the GUI components and be displayed to the end user in a comprehensive way.
This starts in the originating device and then both flows -

value and meta data - meet at the recipients device activating the counterpart plug-ins to record everything accordingly and validate the transaction.

===

image::https://raw.githubusercontent.com/bitDubai/media-kit/master/Readme%20Image/Background/Front_Bitcoin_scn_low.jpg[FermatCoin]

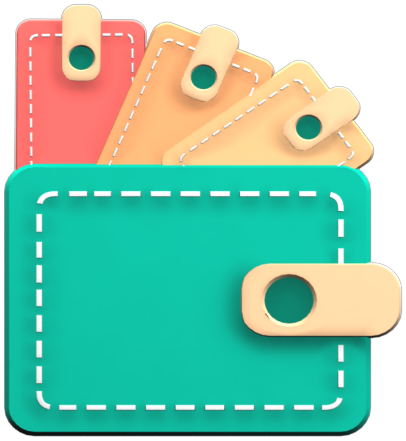
===

Continue Reading ...

Crypto Currency Platform

Next Chapter

Previous Chapter



Fermat

Chapter 11: Fermat Digital Assets Management

What is a Digital Asset ?

A Digital Asset is a digital representation of a product or service that can be obtained in exchange of the asset. Another way of saying this, is that it is a representation of something of value that comes with a "right to use" for the assets possessor. (The asset can be **redeemed** and the end user gets the product or service represented by it)
To create a digital asset two main components are needed:

the asset meta-data

a set of information of what is represented by the asset and the conditions attached to it (**asset contract**)

the validation code

a method that proofs that the asset is genuine and that it

cannot be duplicated or forged+ Fermat incorporates the management of Digital Assets over crypto, being the crypto component what ensures it authenticity among other very useful features granted by using blockchain technology.
The application of Digital Assets is as wide as the digital world. There are some assets types that are familiar and some that sound like sci-fi.

Some types and sub-types

Digital Assets used in Marketing: Coupons and Discounts

Vouchers for services: Tourism vouchers, Mobile Phone Vouchers (Mobile Phone Pre-paid Credit Refill Cards)

Corporate world: Company's shares

Fermat Digital Assets

Other Digital Assets over crypto implementations available in the market are dependent on the cryptocurrency chosen (Bitcoin, Litecoin, Ripple, etc) but thanks to Fermat infrastructural connectivity to several cryptonetworks, Fermat Digital Assets can be issued in different cryptocurrencies. The whole set of functionalities to i

Digital Assets handling in Fermat is provided by *Fermat Digital Assets Platform*, and from it a new whole branch on *platforms based in digital assets* is created. The first application is taken by *Fermat Marketing Platform* which focuses on applications for the Marketing Agencies world.

Continue Reading ...

Digital Assets Platform

Marketing Platform

Next Chapter

Previous Chapter


Fermat CCP
Crypto Currency Platform

Chapter 12: Fermat Crypto Currency Platform

This platform introduces the first **niche wallets** to manage existing crypto currencies - Bitcoin Wallet, Litecoin Wallet, Ripple Wallet, etc - and also the resources to develop wallets of new cryptocurrencies invented in the future.

The new identities introduced in this platform are:+

Intra-Crypto-Wallet-User identity

+

Extra-Crypto-Wallet-User identity

+

For a **niche wallet** to run, we need *reference wallets* and underlying plug-ins which grant:

1. user identification
2. device identification
3. cryptonetwork services

4. cyptoaddress management (address-book)
5. network services
6. world services (to access exchange rates, non-Fermat wallets, etc ...)
7. transactions management

New components

Some of the new components added to the framework by this platform are those to offer the following functionalities:

- crypto address management, crypto money requests and crypto money transmission
- consulting of crypto currency indexes
- crypto money transactions
- Bitcoin Wallet
- Bitcoin Loss Protected Wallet

Bitcoin Wallet

Fermat's Bitcoin **Niche Wallet** is a *basic wallet* in terms of

its functionalities, even simpler as other popular BTC wallets on the market. This is so to maximize its ulterior reutilization.
This wallet provides the minimum necessary functionalities to perform a successful transaction of Bitcoins:

- to make a payment in BTC (send value to a BTC Wallet owner)
- to receive payments in BTC
- to show the transactions
- to show the balance of the wallet
- to show a basic contacts list (user-name and cryptoaddress related to each contact)

NOTE	the simplicity of this wallet is maximized, it does NOT handles payment or money request and and it is *whether* interconnected with other wallets *not* showing anything else but plain BTC currency. (no FIAT money abstraction!)

Bitcoin Loss Protected Wallet

This is the second **Niche Wallet** developed by Fermat, and its main goal is to protect the user from losing money when attempting to spend Bitcoin at a change curse lower that the price when that Bitcoin amount was purchased (entered the wallet).
The new functionalities added are:

- to store the **exchange rate** when a BTC amount is received in the wallet
- to check the **exchange rate** when the user attempts to spend an amount of BTC and to freeze the funds if they are below the **purchase price**

Continue Reading ...

Next Chapter

Previous Chapter

Chapter 13: Fermat Crypto Commodity Money

This platform introduces the first abstraction on crypto currency: as its value is exclusively determined by the market, it can be taken as any other available **commodities** to establish a correlation between the crypto value and the value of any set of Fiat Money that the end user is used to operate with. By means of this, the value is held in cryptocurrency but the transactions and the general wallet functions are displayed in Fiat Currency to make it easier for the end user to run his business in a way he is more familiar with.

New components

Some of the new components added to the framework by this platform are :

- money requests and money transmission registered to show value in **fiat money**
- Crypto Commodity Money Wallet

- Discount Wallet

Crypto Commodity Money Wallet

This niche wallet performs the transaction in crypto currency but the results are displayed in **fiat currency**.

Discount Wallet

This niche wallet aims to protect the value stored in the user's by suggesting profitable conditions to spend it. It checks the *exchange rate* at the time the end user consider ates spending the money and compares it with the rate when the crypto value was purchased, and shows the **positive difference** (gain) as a virtual **discount rate** to the end user at the time he consider ates spending it.

Continue Reading ...

Next Chapter

Previous Chapter


Fermat BNP
Bank Notes Platform

Chapter 14: Fermat Bank Notes Platform

This platform introduces the connection to the physical representation of fiat money, the *bank notes*.
At first hand, it serves as a digital registration in a wallet component of an end user's available stock of money in bank notes that he needs to keep track of for future reference. This leads to build a traceability of these funds when participating in future transactions.

Bank Notes Wallet

Continue reading ...

Next Chapter

Previous Chapter


Fermat SHP
Shopping Platform

Chapter 15: Fermat Shoping Platform:

This platform is designed to allow the interaction between the actors of traditional consumer market of products and services.

The new identities introduced are:

Shop

to identify shop owners who accept cryptocurrency in payment of their products

Brand

for brands who accept cryptocurrency in their retailers network

Retail

for retailers of brands that accept cryptocurrency

New components

Some of the new components added to the framework by this platform are those to offer the following functionalities:

- Purchase Transmission (purchase of products payed with crypto currency)
- Purchase and Sale transactions
- Wallets: Shop Wallet, Brand Wallet and Retail Wallet

Shop Wallet

This wallet includes the specific new functionalities:

- to receive payments in crypto currency
- to connect a purchase bill of the shop with a crypto request to cover the fiat currency amount billed

Brand Wallet

Retail Wallet

Continue reading ...

Next Chapter

Previous Chapter


Fermat CsH
Cash Platform

Chapter 16: Fermat Cash & Fermat Bank Platforms

These two platforms form the basic infrastructure to interact with the legacy financial system.

New components

The new components added to the framework offer the following functionalities:

Cash Money transactions

: to keep track of transaction where cash money is involved.

Bank Money transactions

: to keep track of transaction where bank money is involved.

Cash Money Wallet

: only a **reference wallet** to provide services to other high level wallets by holding the accountancy of the end

user's amount of cash.

Bank Money Wallet

only a **reference wallet** to provide services to other high level wallets by holding the accountancy of the end user's amount of money in an specific bank account.

Continue reading ...

Next Chapter

Previous Chapter


Fermat CBP
Crypto Broker Platform

Chapter 17: Fermat Crypto Broker Platform

In this platform the actors involved are the **crypto broker** and his customers (**crypto customers**). A **crypto broker** is a person or small organization whose business logic is to buy and sell cryptocurrency for fiat money in his local market, obtaining his profits from the difference in the buy/sell price. We distinguish also two usual levels of **crypto brokers**, those acting as **crypto brokers wholesalers** and those who deal with final customers. The key goal of this platform is to **facilitate** the most of the routine tasks to run the business and even provide the necessary functionalities to prevent *always* the loss of money in every transaction.

New components

The new components added to the framework offer the following functionalities:

Fiat Index

to check the exchange rate of fiat currencies.

Business transactions

to handle more complex transactions that include negotiation, contracts (agreements), sale and purchase of crypto and fiat money.

Crypto Broker Platform Wallet Manager

to manage different wallets according to multiple actor identities in the three actor types (as crypto broker, cryptocustomer, crypto wholesaler).

Community Components

to keep track of relationships between Crypto Brokers and Crypto customers by defining communities.

Wholesalers

contact list of available crypto wholesalers.

Crypto Broker Wallets

Crypto Broker Wallet , Crypto Broker Customer Wallet

Fermat Crypto Broker Wallet

This Wallet is designed with the theoretical assumption that

the broker always has available stock of cryptocurrency and fiat money and that his activity is to be constantly selling this two kind of products obtaining profit. The functionalities offered should help him to run the business in a simple way, taking the responsibility of the basic management of the broker's fund (in cryptocurrency and fiat money, in cash and in bank accounts), the management of the contact list of his customers, and a balance of the profits generated.
Key functionalities are:

- **Funds restock and destock**:
  - Crypto Currency
  - Cash Money (Fiat money in cash)
  - Bank Money (Fiat money in a bank account)
- **Incoming/Outgoing money**:
  - Cash on Hand
  - Delivered Cash
  - Money Bank Deposit
- **Special functions** for *associated Crypto Brokers network* (**agreement** between colleagues to be able to manage

joint operations when needed) :

- keep track of **Broker-to-Broker contract**

Fermat Crypto Customer Wallet

It has the complementary functionalities of the transactions involved in dealing with a Crypto Broker, keeping track of the details how each exchange was actually made.

Continue reading ...

Next Chapter


Fermat CDN
Crypto Distribution Network

Chapter 18: Fermat Crypto Distribution Network

Developing Regional Crypto Ecosystems

In the actual stage of development of the crypto currency industry, there is still a very restraint availability of crypto currency due to the reduced circulation regarding the transaction volume and the number of actors involved in the business. Fermat implementation brings to the market a lot of tools and products to facilitate the exchange of crypto currencies in various ways. A key point in building a crypto distribution network is the role of the cryptobrokers whose business logic is straightforward the constant selling of crypto and fiat money. However, to extend the cryptobroker activities to a larger network of interactions that can build a **local** circulation of crypto currencies, the functionalities

covered by the previous *Crypto Brokers Platform* are not sufficient. This new platform adds what is necessary to achieve this goal.
There is a new set of circuits where the crypto and fiat money flow, and several complexities to be taken into account. The new components not only serve to deal with this new interactions but also motivates the adoption of new roles for the different actors stimulating a web of interconnections where value is always being transmitted within a local environment of end users. Thus, a regional crypto ecosystem is created!

New Identities

In order to complete the scheme of a *Crypto Distribution Network* these actors were added:

Crypto Wholesaler

: organizations owing a certain large amount of crypto-credit to sell through a distribution chain.

Crypto Distributor

: the next level in the distribution chain, buying to wholesalers and re-selling to Top Up Point

Top Up Point

shops/organizations/persons who transfer crypto-credit in exchange for fiat-money **(receives money - gives crypto)**

Cash Out Point

shops/organizations/persons who gives out money in exchange for a transfer of crypto **(receive crypto - gives money)**

New Components:

These identities add a considerable new sets of interactions that are covered by the new components.

Business transactions

new transactions types between the new actors, and new contracts.

Wallets

Crypto Wholesaler Wallet , Crypto Distributor Wallet, top Up Point Wallet and Cash Out Point Wallet

Crypto Wholesaler Wallet

Crypto Distributor Wallet

Top Up Point Wallet

Cash Out Point Wallet

Continue reading ...

Next Chapter

Previous Chapter


Fermat DAP
Digital Asset Platform

Chapter 19: Fermat Digital Asset Platform

Blockchain beyond currency

With the adoption of *Digital Asset Management* within Fermat, the framework establishes two distinct branches and its interconnected platform sets, one managing cryptocurrencies and therefore dealing with different cases of money transactions and another one, starting with this platform, to manage transactions and services based on digital assets.

Asset Characteristics

1. They are created by an **asset issuer**
2. They have an **expiration date**
3. They belong to a certain **asset type** and **asset sub-type**
4. They have an associated **meta-data** (image, sound, multimedia file, etc)
5. They have compulsory conditions to comply enclosed in

the **asset contract**, related separately to the **issuer**, to its **type** and **sub-type**

New Identities

The new identities introduced to run the basic life cycle of an digital asset are:

Asset Issuer

creator and owner of the asset. He "hashes" the assets meta-data and link it to a *crypto transaction* of a small crypto currency amount to a predefined **asset issuer crypto address**, and by this conjunct actions the asset is **issued**.

Asset User

is the original recipient of the asset. After receiving it (as a part of a distribution campaign for example), he can perform several actions later described in the available transactions for digital asset, but the most frequent could be the transfer to a *redeem point* to exchange it for the product/service represented by it.

Redeem Point

is the point where the asset is exchanged for the service/product.

New Components

The business logic of the use of digital assets and the new interrelations between the actors involved need the following services to be granted:

Wallets

one for each actor: Asset Issuer Wallet, Asset User Wallet, Redeem Point Wallet.

Transactions

Issuing, Distribution and Reception of digital assets.

Issuing

tools to create an asset: *Asset Factory*

Redemption

three possible ways: by the Issuer, by the User and by a Redeem Point.

Appropriation

possibility of keeping the asset and **appropriating** the

inherited crypto currency amount (transforming it into its **nominal value**)

Community

tools to linking the Digital Assets actors among them (Issuer, User and Redeem Point Communities)

Continue Reading ...

Fermat Digital Assets Management

Next Chapter

Previous Chapter


Fermat MKT
Marketing Platform

Chapter 20: Fermat Marketing Platform

Marketing with Digital Assets

The first implementation for Digital Assets aims to provide the Marketing Sector with a valuable tool for medium to large size enterprises to run their campaigns profiting from the advantages of this technology. Assets take form of Vouchers, Coupons and Discount Offers commonly used in marketing strategies.

New Actor

Marketer

> responsible of elaborating marketing strategies in marketing areas of large enterprises or as a member of independent marketing agencies providing services to the corporate world.

New Components

Transactions

incoming and outgoing assets types.

Wallet Branding

a special **Wallet Editor** to re-brand a niche wallet into a new one by changing their *navigation structure* and _GUI components (skins..) with an intuitive interface **without** programming but **re-utilizing** the functionalities provided by Fermat wallets.

Wallets

to store and manage Vouchers, Coupon, Discounts...

Continue Reading ...

Next Chapter

Previous Chapter

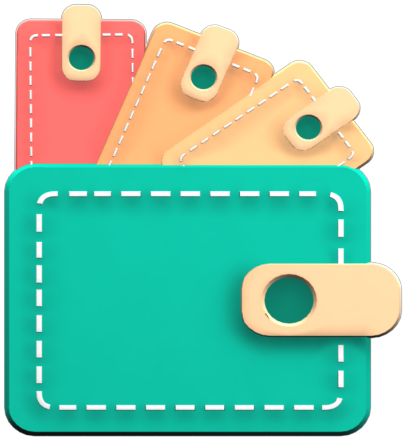
Chapter 21: Fermat Digital Private Network

Securing your funds

Continue Reading ...

Previous Chapter

Back to Book Intro



Fermat

Quick Glossary

This quick glossary contains many of the terms used in relation to **Fermat**, **blockchain technology** and **cryptoworld**.

These terms are used throughout the book, so bookmark this for a quick reference.

address (aka crypto address)

> A crypto address looks like 1DSrfJdB2AnWaFNgSbv3MZC2m74996JafV. It consists of a string of letters and numbers starting with a "1" (number one). You use an cryptoaddress to send others cryptocurrency or to receive it in your wallet.

block

> A grouping of transactions, marked with a timestamp, and a fingerprint of the previous block. The block header is hashed to produce a proof of work, thereby validating the transactions. Valid blocks are added to the main blockchain by network consensus.

blockchain

A list of validated blocks, each linking to its predecessor all the way to the genesis block.

blockchain technology

crypto network

A peer-to-peer network that propagates crypto transactions and blocks to every node runninmg a specific crypto currency protocol.

secret key (aka private key)

The secret number that unlocks the cryptocurrency sent to the corresponding crypto address. A secret key looks like 5J76sF8L5jTtzE96r66Sf8cka9y44wdpJjMwCxR3tzLh3ibVPxh.

transaction

In simple terms, a transfer of crypto currency from one address to another. More precisely, a transaction is a signed data structure expressing a transfer of value. Transactions are transmitted over the crypto network, collected by miners, and included into blocks, made permanent on the blockchain.

wallet

Software that holds all your crypto addresses and secret keys. Use it to send, receive, and store your crypto currency.

Index

{toc}

Unresolved directive in 1-fermat-book.asciidoc - include::book-appendix-01-principles.asciidoc[]

Unresolved directive in 1-fermat-book.asciidoc -

include::book-appendix-02-history.asciidoc[]

Unresolved directive in 1-fermat-book.asciidoc - include::book-appendix-03-money.asciidoc[]