# Ermis Protocol Whitepaper

Shresth Agrawal, Nikolas Kamarinakis, and Orfeas Stefanos Thyfronitis Litos

**Abstract.** NFTs have revolutionized the way digital art is created and admired, ushering in a new age for artists and fans alike. This transformation has transcended the digital domain and entered the physical one via *phygital tokens*, or simply *phygitals*. These bridge the gap between the two domains by tying an NFT with a physical item.
Through the Ermis protocol, Ēnosys enables the secure *redemption* of a phygital in exchange for its associated item with minimal trust. Ermis provides phygital redemption by tapping into the assurances of the Flare blockchain, allowing for the first time vendors and buyers to connect directly via an easy-to-use smart contract. Ēnosys is not involved in the physical delivery and does not claim any ownership in the digital domain at any point.

## 1 Introduction

The world of art has been irrevocably changed by the advent of *non-fungible tokens* (NFTs). These enable the ownership and sale of digital art via blockchain, tearing down the limitations of the physical world and disintermediating the process of art sale by removing the need for legal contracts and art galleries.

However, due to the strictly digital nature of blockchains, securely tying a physical item with an NFT (a.k.a. *phygital token*) is not straightforward and, to the best of our knowledge, requires an intermediary that is, to some degree, trusted. Minimizing this trust while providing a practical, dependable protocol is a venerable goal, which the Ermis Protocol achieves.

In particular, the Ermis Protocol addresses a crucial step of the lifecycle of a phygital: it allows the owner to redeem (which usually, but not always, means to burn) it on-chain and get the corresponding physical item delivered by the vendor – all without implicating Ēnosys[1]. The latter is only involved earlier: when the vendor is onboarded, Ēnosys performs KYC and requires proof that the item is available to ship.

Ēnosys is an already established party in the blockchain space. It is therefore well positioned to offer and support the Ermis protocol.

## 2 High Level Overview of Ermis

### 2.1 Before the Protocol

To set the stage for the Ermis protocol, two things must have happened ahead of time. First, the Provider (e.g., the artist) has to have performed KYC with

[1] https://enosys.global/

Ēnosys and furnished proof that they have and can deliver the physical item. A picture or video of the item is sufficient. This is needed to prevent fraudulent or illegal usage, as delivery of the physical item cannot be enforced on-chain. The Provider must trust that Ēnosys will carefully guard its private data and only disclose it in case of fraud by the Provider. Initially, becoming such a provider will only be available via submission to the Ēnosys team, but the plan is to eventually open Ermis to external vendors.

Second, the Provider must have sold the phygital. Since they are NFTs, phygitals are tradeable on-chain exactly like any other NFT. As it is a well-established kind of transaction and independent of Ermis, we will not go into detail on how a phygital can be sold.

As a result, a potential buyer that wants to own the physical item without having it at hand (e.g., an art dealer) will be content with having just the phygital. The phygital may change many hands many times until it is bought by someone that desires the corresponding physical item (e.g., a fan of the artist).

## 2.2 Honest Protocol Flow

We will now go over the protocol flow when everything works as intended (see Figure 1). All communication happens via on-chain transactions. The relevant messages are emphasized. First, the phygital Redeemer *opens* a request to redeem the phygital, including its postal address in the request, encrypted for the Provider. Then the Provider decrypts and checks the address. If it is valid, they *accept* the request. At that point a countdown starts, within which the Provider has to *fulfill* the request. Indeed, the Provider ships the physical item to the Redeemer's address, and does four actions atomically: informs the Redeemer that the request was fulfilled, provides encrypted tracking information, burns the phygital and creates a Soul-bound Token [?] under the possession of the Redeemer. Eventually the item is delivered to its Redeemer. With this straightforward protocol, the redemption is successful. No interaction with Ēnosys whatsoever has been necessary.

## 2.3 Benign Failure Scenarios

Let us now examine the various ways in which redemption can fail. We will first discuss mishaps after a redemption request is placed, but before it is accepted. If the Provider is unable to ship or in case the postal address in the redemption request is malformed, the Provider cannot fulfill and thus has to *reject* the request. The phygital returns to the Redeemer.

It is also possible that the Provider does not accept for a long time or that the Redeemer changes its mind soon after placing the request. In either case and as long as the Provider has not yet accepted, the Redeemer can *close* the request. Once again, the phygital returns to the Redeemer.

Now we will focus on the case of a successful request and accept, but failure to fulfill. In this scenario, the countdown (which had started when the Provider

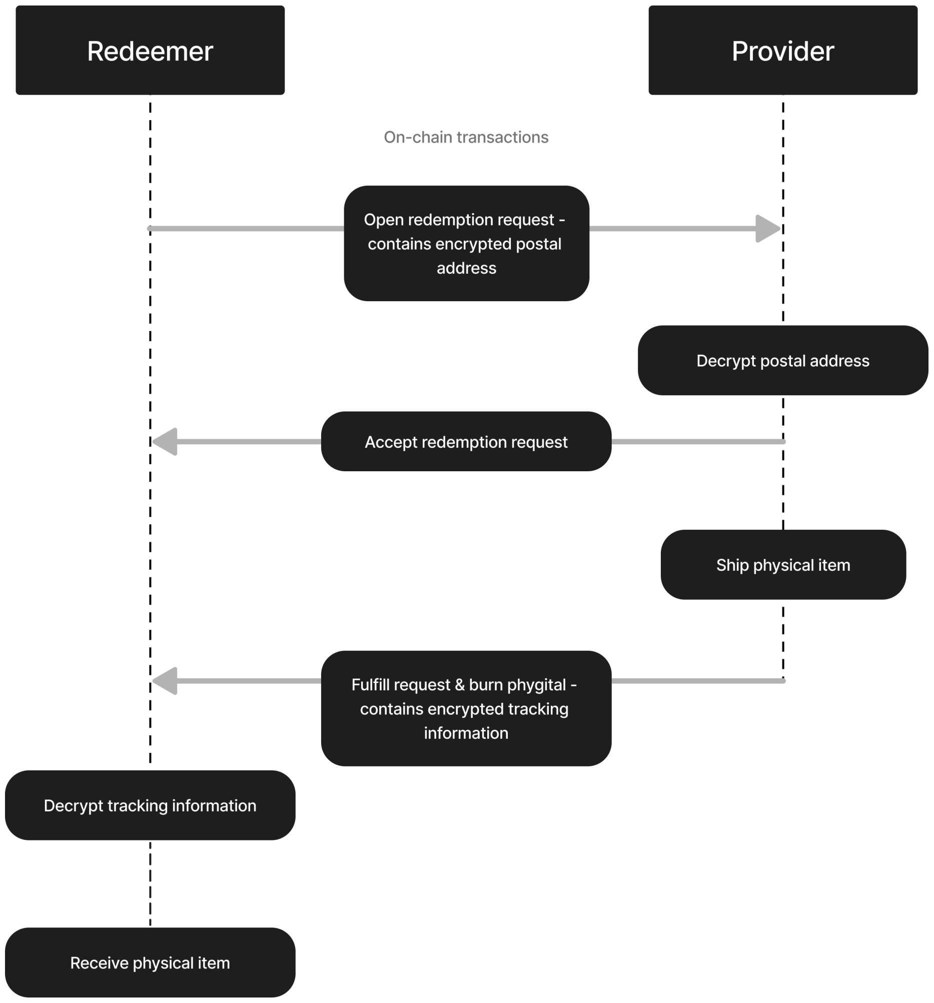
Redeemer
Provider
On-chain transactions
Open redemption request - contains encrypted postal address
Decrypt postal address
Accept redemption request
Ship physical item
Fulfill request & burn phygital - contains encrypted tracking information
Decrypt tracking information
Receive physical item

**Fig. 1.** Ermis Protocol

accepted) will eventually reach zero. At that point the Redeemer can *expire* the request. Just like in the two previous cases, the phygital returns to the Redeemer.

In all three previous failure scenarios no harm has been done, except possibly for some wasted transaction fees. Indeed both parties are back at their initial state: the Redeemer has the phygital and the Provider has the physical item.

### 2.4 Disputes

Unfortunately, there is another, arguably much more problematic, failure scenario: The Provider sends a fulfill transaction without actually shipping the physical item, thus defrauding the Redeemer by burning the phygital while keeping the physical item.

Since the blockchain is confined to the virtual realm, it is not straightforward, and likely impossible, to enforce delivery or revert the destruction of the phygital in such a case without the help of a trusted third party. This is where the initial KYC by Ēnosys becomes relevant: In case of a dispute, either party can raise the issue to Ēnosys and pursue adjudication with their aid or even via traditional legal avenues. Since all communication is visible on-chain and all messages are signed, the entire transcript can be provided to Ēnosys with guaranteed authenticity. The exact dispute resolution mechanism will be covered in later revisions of this document. Both the Redeemer and the Provider have to trust that, as long as they behave honestly throughout the Ermis protocol and the adjudication process, the dispute will be resolved in their favor by Ēnosys.

### 2.5 Trust Assumptions

We here reiterate the ways in which the two parties have to trust Ēnosys for clarity.

- The *only* way in which the Redeemer has to trust Ēnosys is to resolve any disputes to their favor as long as the Redeemer behaves honestly.
- The Provider has to trust Ēnosys in two ways:
  - Like the Redeemer, the Provider has to trust that, as long as it is honest throughout the protocol and fully cooperative with any dispute adjudication process, Ēnosys will resolve all disputes to its favor.
  - The Provider must trust that Ēnosys will manage the data gathered at KYC with the required care: As long as the Provider is honest, their private data will be stored securely by Ēnosys, only shared with the minimum number of parties needed and thoroughly deleted once they are not useful anymore.

The reputation that Ēnosys has accumulated in the blockchain and web3 space indeed justifies bestowing this minimal level of trust.

## 3 Smart Contract Details

Consider a phygital $p$ (conforming to ERC-721[2] or ERC-1155[3]) owned on-chain by redeemer *Alice* and the corresponding physical item $g$ (be it a piece of art, a consumer good, a collectible, etc.) held by provider *Bob*. The goal of the protocol is *phygital redemption*, i.e., that *Alice* burns $p$ on-chain and *Bob* ships $g$ to her physical address and gives her a matching soul-bound token in exchange.

At the heart of this novel protocol is a set of Solidity Smart Contracts[4] developed in collaboration with Common Prefix[5]. The main contract is called `PhygitalRedeemer721` or `PhygitalRedeemer1155` depending on the type of the phygital and inherits from `PhygitalRedeemerCommon`. We call it `PR` from now on. Each phygital corresponds to exactly one `PR`, which is created by Ēnosys via a call to the `createRedeemer()` method of its long-lived `PhygitalRedeemerFactory` contract when the phygital is first created. Upon the creation of `PR`, the aforementioned countdown, a.k.a. "maximum fulfillment duration" (which prevents the Provider from delaying fulfillment indefinitely) is also set. It cannot be changed later. A state machine of `PR` can be seen in Figure 2.

To initiate redemption, *Alice* calls `PR.openRequest()`, specifying her public key, the ID of $p$ and her encrypted shipping address. A `requestID` is returned. As an optimization, a single `PR.openRequest()` can be used to request multiple phygitals to be redeemed at once using the exact same protocol, but we will here focus on a single phygital for simplicity.

After the request is opened there are three possibilities:

– *Alice* changes her mind and calls `PR.closeRequest()` with the aforementioned `requestID`,
– *Bob* declines the request by calling `PR.rejectRequest()` with the `requestID` and a free-form reason for rejecting. Possible reasons are, e.g., an invalid shipping address,
– *Bob* accepts the request by calling `PR.acceptRequest()` with the `requestID`.

In the first two cases, the contract returns to its initial state and $p$ returns to *Alice*. In the third case, $p$ is held by the contract and a countdown equal to the maximum fulfillment duration starts.

Within that time, *Bob* has to ship the physical item and call `PR.fulfillRequest()` with the `requestID` and a free-form field containing encrypted delivery tracking information. This function stops the countdown, burns the phygital and creates a soul-bound ERC-721 token owned by *Alice*.

If the countdown runs out before fulfillment, *Alice* can then cause expiry by calling `PR.expireRequest()` with `requestID`. In this case the contract returns once again to its initial state and $p$ returns to *Alice*. If *Alice* does not expire the request, it is still possible for *Bob* to fulfill, even after the countdown is over.

[2] https://ethereum.org/en/developers/docs/standards/tokens/erc-721/
[3] https://ethereum.org/en/developers/docs/standards/tokens/erc-1155/
[4] https://github.com/flrfinance/phygital-redemptions-contracts
[5] https://flrfinance.medium.com/introducing-fflabs-cfa8d580441a

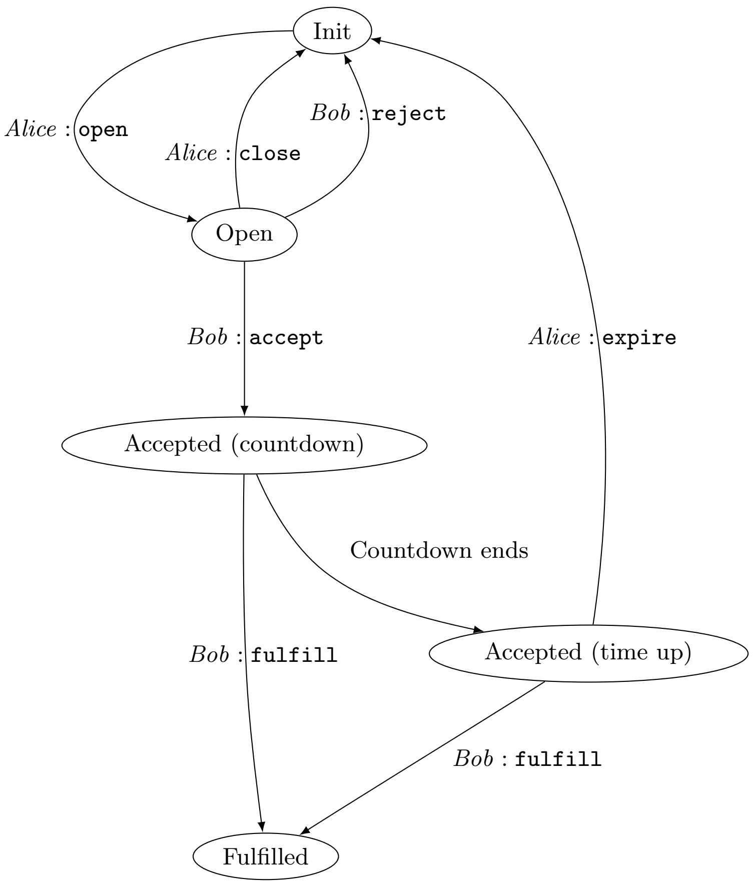

**Fig. 2.** State Machine for Phygital Redemption Request

*Alice* can decrypt and use the tracking information to monitor the progress of the physical item delivery, which should eventually arrive. The protocol is now complete.

In case *Bob* calls `PR.fulfillRequest()` but does not actually ship the physical item, *Alice* should open a dispute with Ēnosys. The exact procedure remains as of now unspecified.

### 3.1 Alternative design with one transaction per party

One of the design choices of Ermis is that the provider submits two transactions: one for accepting and one for fulfilling a request. This choice is made for a number of user experience-enhancing reasons: Firstly, it ensures that the countdown starts only after the provider first becomes aware of the request, thus avoiding a situation in which the request is opened, the countdown almost completes and only then does the provider check the contract – this can lead to the provider rushing to fulfill and worsens their user experience. Secondly, the redeemer gets more fine-grained feedback on the progress of their request. Lastly, this design ensures that all communication happens on-chain, which both simplifies the user experience and increases accountability.

The drawback of this design is increased on-chain costs. To avoid this, `acceptRequest` and `fulfillRequest` can be combined in one message. In that case, the countdown must start when the request is opened (and thus the default maximum fulfillment duration should be higher) and the physical item should only be shipped after `fulfillRequest` is finalized (since it is in a race with a possible `closeRequest` by the redeemer). Since tracking information is generated only after shipping begins, it has to be sent to the redeemer out-of-band. To achieve this, the redeemer can include an encrypted social media handle in its call to `openRequest`. This alternative can be easily implemented if popular. If the out-of-band communication is non-repudiable and usable by arbitrators in case of dispute, this alternative design provides exactly the same guarantees as the original.

## 4 Future Work

A number of tasks are deferred to future iterations.

- As discussed in Subsection 2.4, dispute resolution will be covered in later revisions of this document. As Ēnosys understands and is cognisant of potential risks, they want to outline their intention to restrict Ermis as an internal tool until the dispute resolution mechanism has been implemented.
- Minting redeemable NFTs will not be available to external vendors upon launch. The Ēnosys team will explore initial use cases before B2B integrations occur. Fees will be specified and applied to vendors in later protocol versions.

- In later versions of Ermis, the user will also be able to choose to collect their item from select designated locations around the world. A fee would be applied in such cases.
- As discussed in Subsection 3.1, it is possible to merge the *accept* and *fulfill* transactions to reduce on-chain fees for the Vendor, albeit at a small expense of usability. The implementation of this alternative will be considered if the on-chain fees are deemed too high.
- Redemption now is tied to burning the phygital. This, however, may not fit all use cases. In the future, arbitrary redemption logic (defined upon the creation of the phygital by its creator) will be enabled. Here are two possible examples:
  - The phygital is not burned, but kept by the Redeemer.
  - Multiple phygitals are required to open a redeem request but only one of them is burned.
- Ēnosys will offer escrow services for the physical item. If a Redeemer does not trust the Provider to ship the item but instead trusts Ēnosys for that, Ēnosys could first receive the item, signal the Redeemer and only then would the latter open the request.

## 5 Conclusion

We have here presented the Ermis Protocol, which enables the redemption of phygitals. Via this protocol, parties can obtain a physical good that corresponds to a phygital they own. In the optimistic case, the whole process is completed in private, i.e., with no interaction with Ēnosys if both parties are honest. A minimal amount of trust towards Ēnosys is needed to ensure correct, effective and timely dispute resolution in case one party attempts fraud.

We believe that the Ermis Protocol successfully bridges the virtual with the physical world in a cheap, secure and effective manner, opening new, exciting avenues for people to interact.