〈Obligations Regarding SIVIRA Inc. Confidential Information 〉 You agree to protect SIVIRA Inc. Confidential Information using at least the same degree of care that You use to protect
Your own confidential information of similar importance, but no less than a reasonable degree of care. You agree to use SIVIRA Inc. Confidential Information solely for the purpose of
exercising Your rights and performing Your obligations under this Agreement and agree not to use SIVIRA Inc. Confidential Information for any other purpose, for Your own or any thirdparty's benefit, without SIVIRA's prior written consent. You further agree not to disclose or disseminate SIVIRA Inc. Confidential Information to anyone other than: (i) those of Your
employees and contractors, or those of Your faculty and staff if You are an educational institution, who have a need to know and who are bound by a written agreement that prohibits
unauthorized use or disclosure of the SIVIRA Inc. Confidential Information; or (ii) except as otherwise agreed or permitted in writing by SIVIRA Inc.. You may disclose SIVIRA Inc.
Confidential Information to the extent required by law, provided that You take reasonable steps to notify SIVIRA Inc. of such requirement before disclosing the SIVIRA Inc. Confidential
Information and to obtain protective treatment of the SIVIRA Inc. Confidential Information. You acknowledge that damages for improper disclosure of SIVIRA Inc. Confidential Information
may be irreparable; therefore, SIVIRA Inc. is entitled to seek equitable relief, including injunction and preliminary injunction, in addition to all other remedies.
The information herein is for informational purposes only and represents the current view of SIVIRA Inc. as of the date of this presentation.
Because SIVIRA Inc. must respond to changing market conditions, it should not be interpreted to be a commitment on the part of SIVIRA Inc., and SIVIRA Inc. cannot guarantee the
accuracy of any information provided after the date of this presentation.
SIVIRA Inc. MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.

Contract Design

本稿では、unWallet を構成するコントラクト群の全体構造と、ウォレットの基本的な機能を担う Module Contract の挙動について説明する。

1. 全体構造

1-1. 概要

図 1 に示すように、unWallet は複数のコントラクトから構成されている。まず、これら主要コントラクトの概要を以下に記載する。

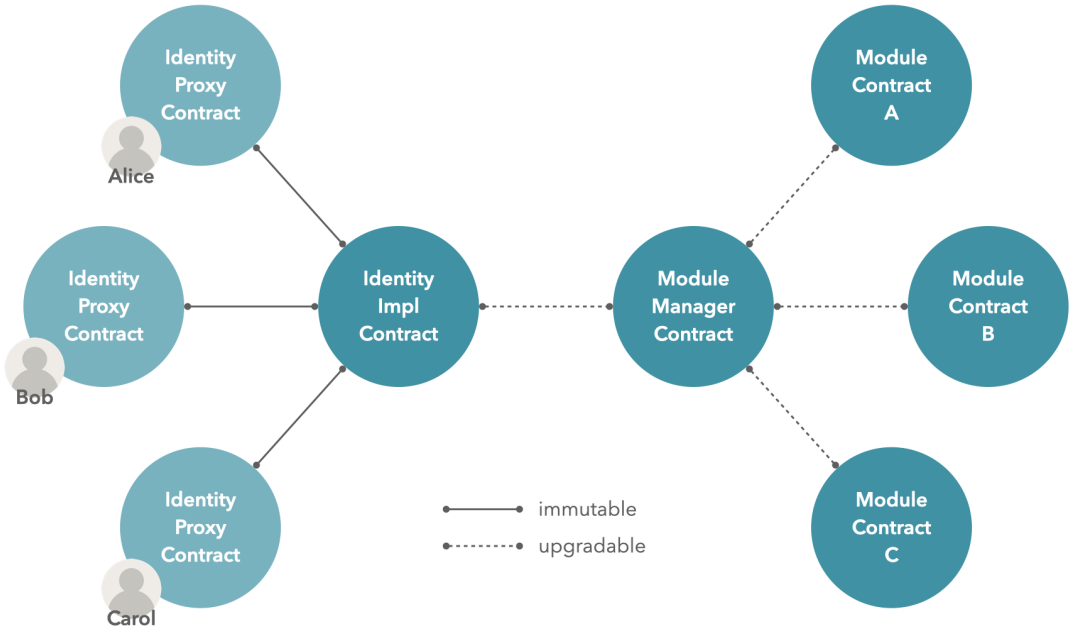

図 1. 主要コントラクト群

1-1-1. Identity Proxy Contract

エンドユーザーごとに生成される、非常に軽量なプロキシコントラクト。このコントラクトのアドレスが各エンドユーザーのウォレットアドレスとなる。その名の通りプロキシであるため、全ての処理は delegatecall を介して Identity Impl Contract に委任される。

1-1-2. Identity Impl Contract

ウォレットオーナー変更や外部コールなど、ウォレットのコア機能のみが実装されたコントラクト。ほぼ全ての機能は Module Contract を介してしか利用することができないため、上記のような処理を実行するためには各 Module Contract に実装された検証処理を通過する必要がある。Module Contract からのアクセスの制御は、このコントラクトに設定された Module Manager Contract を基準に行う。

また、このコントラクトに対する任意の関数コール（onERC721Received など）をあらかじめ指定しておいた Module Contract に委任する機構なども有する。なお、この指定も Module Manager Contract にて行う。

1-1-3. Module Manager Contract

Identity Impl Contract に対してアクセス可能な Module Contract を管理するコントラクト。このコントラクトに登録された Module Contract のみが、Identity Impl Contract に実装されたコア機能を利用できる。

何らかの事情により全ての Module Contract が登録解除され、ウォレットが機能停止してしまうような事態を避けるため、指定した Module Contract をロックし、永続的に登録解除できなくする機構も有する。

1-1-4. Module Contracts

ウォレットの各機能が実装されたコントラクト群であり、各機能のエントリーポイントを担う。何らかの検証処理（入力された電子署名がウォレットオーナーによるものかどうかの確認など）を行った上で、最終的には Identity Impl Contract に実装されたコア機能を稼働させる、というのが基本的な挙動である。

Module Contract が実装する機能の例を以下に示す。

- 特定の条件を満たした場合のみ Identity Proxy Contract から外部コールを実行するメタトランザクション機能
- 特定の条件を満たした場合のみウォレットオーナーの変更を行うリカバリー機能

1-2. 詳細

1-2-1. Upgradability

unWallet は、1-1 で述べたようなモジュール構造を有するゆえに、一般的なプロキシ構造のコントラクトとは異なる形式でアップグレードを行う。具体的には、図 2 のように、Module Manager Contract が Module Contract の登録やその解除を行うことでアップグレードを行う。

なお、このアップグレードは、該当の Module Manager Contract が設定された Identity Impl Contract と接続する Identity Proxy Contract 全てに対して適用されるため、Identity Proxy Contract が大量に生成された場合であっても対応は容易である（が、適用範囲が広いほどリスクも大きくなるため、将来的には適切な単位で Module Manager Contract を切り分けることが重要になるだろう）。

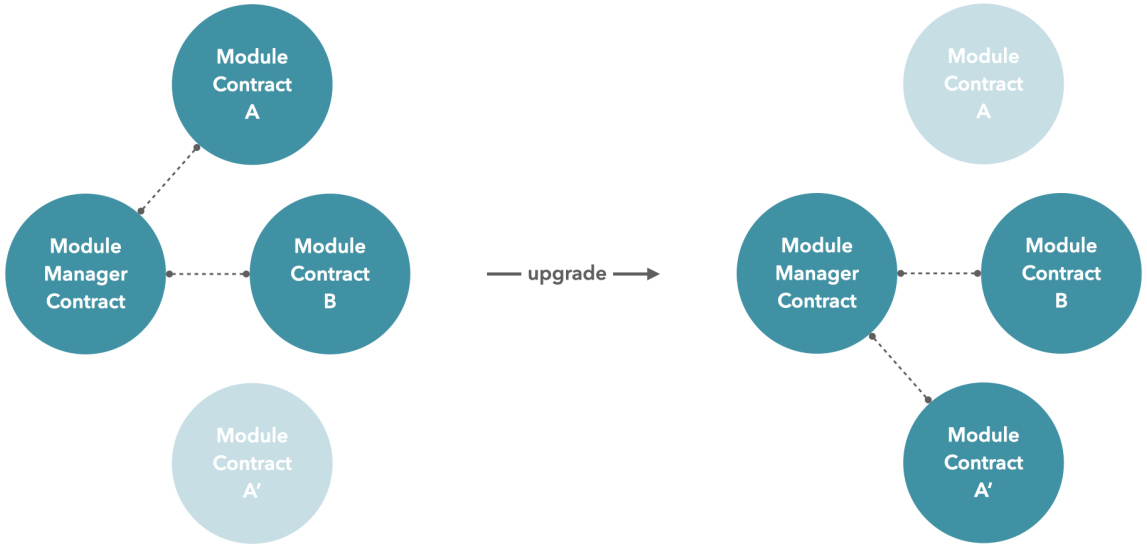

図 2. モジュール構造ベースのアップグレード

一般的なプロキシモデルでは、1 プロキシコントラクトに対して複数の実装コントラクトを設定することができないため、部分的なアップグレードが困難だが、上記のようなモジュール構造であれば、例えば、機能 A を担う Module Contract A のみを Module Contract A' に差し替えることで容易に実現可能である。

なお、Module Contract の登録やその解除は Module Manager Contract のオーナー（単一の EOA もしくはコントラクトアカウント）が自由に行える。緊急度の高いバグ修正や脆弱性解消が発生する可能性が高いだろう初期フェーズにおいては、1-1-3 で述べたような Module Contract のロック機構を活用したり、何らかの制約（Module Contract の登録やその解除を申請してからそれが実行されるまでに一定の時間経過を必須とするなど）を設けたりしながら、開発チームが Module Manager Contract の管理を行う予定だが、中長期的には、何らかのガバナンス機構を

有したコントラクトアカウントを Module Manager Contract のオーナーに設定し、開発チームの権力を徐々に弱めていく予定である。

また、unWallet のモジュール構造は、コントラクトウォレットのような “コールする側” のコントラクトの upgradability に対しても利点がある。具体的には、新旧バージョンの実装を並存させながらの段階的なアップグレードが可能である。これは例えば、Module Contract A をその新バージョンである Module Contract A’ にアップグレードしたい場合、以下のようなフローでそれを行うことができることを意味する。

1. Module Contract A’ を登録する
2. Module Contract A と Module Contract A’ の 2 つが並存した状態でクライアント（ウォレットの UI となるアプリケーション）群の実装対応を待つ
3. 全てのクライアントが Module Contract A’ に対応した後、Module Contrac A の登録を解除する

もちろん、何らかの事情によって Module Contract A を使い続けたいクライアントが存在する場合、「手順 3 を行わない」という選択肢も存在する。

ウォレットは、様々なプロトコルや Dapps が展開されていく中で、その変化に適応しながら、オンチェーンにおける日々のアクションの “主語” として機能する必要がある重要なプロダクトであるため、unWallet は、ここまでで述べたような柔軟な upgradability を備えている。

なお、トークンコントラクトのような “コールされる側” のコントラクトの upgradability は、受けたコールの委任先を制御する機構が中心であり、基本的には、その委任先はある時刻において単一であるため、特定のコールに対して上記のように複数バージョンの実装を並存させることはできないが、コントラクトウォレットのような “コールする側” のコントラクトの upgradability は、外部コールを行うまでの経路を制御する機構が中心であり、その経路は（結果的に同じ外部コールを行うとしても）上記のように複数存在しても問題はないし、どの経路を使うかはウォレットオーナーが決めればよい。upgradability を設計する際は、このようなコントラクトの性質に対して適切な機構を選択することも重要である。

2. モジュール

unWallet の基本的な機能を担う Module Contract の挙動について説明する。

2-1. Relayer Module Contract

Identity Proxy Contract から任意の外部コール（メタトランザクション）を実行するための Module Contract。

メタトランザクションの実行を担う関数は、以下のような引数をとる。

- 実行したい外部コールの内容と、それに要した手数料の返金方法を指定したデータ
  - どの Identity Proxy Contract からどのコントラクトのどの関数をどのような引数で実行するのか
  - Relayer EOA（メタトランザクションの起点となる EOA）が支払ったトランザクション手数料を Identity Proxy Contract が返金するかどうか
    - 返金するのであれば、どのトークン（ETH や ERC20 トークンなど）で返金するのか
- 上記データに対するウォレットオーナーによる電子署名
  - 厳密に言うと、署名対象データにはリプレイアタックを防ぐためのデータなども含まれる

これらがこの Module Contract 内での検証を通過できた場合、指定した外部コールが図 3 のようなフローで実行される（返金も外部コールの一種と考えればよい）。

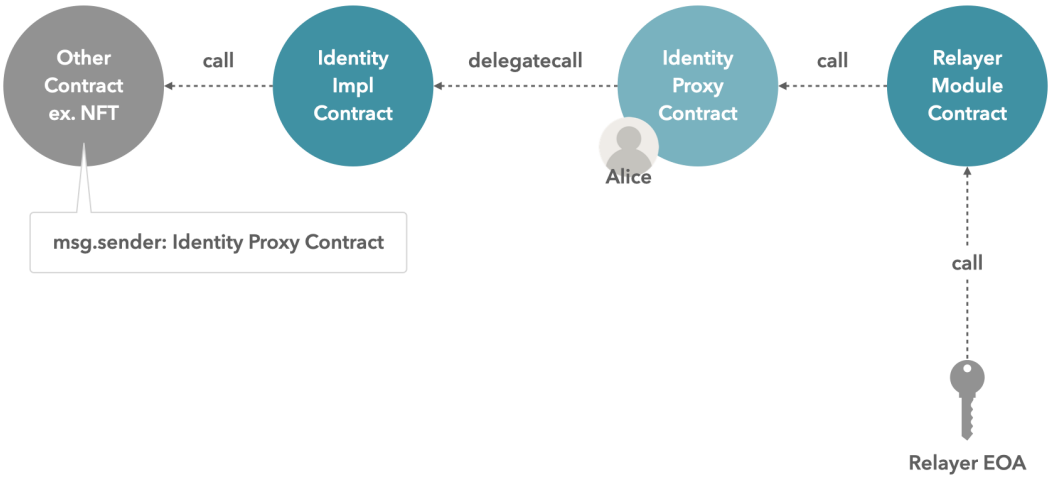

図 3. メタトランザクションの実行フロー

なお、正当な引数を渡すことができるのであれば、どんな EOA でも Relayer EOA になることができる。また、返金を行わないということは、本来はウォレットオーナーであるエンドユーザーが支払うべきトランザクション手数料を Relayer EOA が肩代わりすることを意味する。よって、例えば、何らかの Dapps を提供する組織が、その利用ハードルを下げるために、自身の Dapps に属するコントラクトへの外部コールを行うメタトランザクションに関してのみ Relayer EOA を担い、エンドユーザーの手数料を肩代わりすることなども可能である。

2-2. Delegate Module Contract

Identity Proxy Contract に対する任意の関数コールの委任を受ける Module Contract。

この Module Contract に処理を委任したい関数コール（onERC721Received など）は、その function selector とこの Module Contract のアドレスをあらかじめ Module Manager Contract に登録しておく必要がある。委任を行うように設定された関数コールに関しては、図 4 のようなフローで委任が行われる。

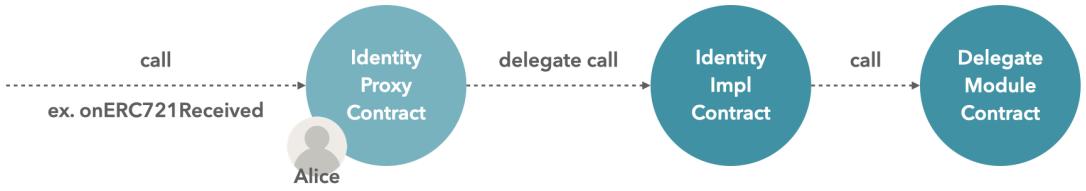

図 4. Identity Proxy Contract に対する関数コールの委任フロー

2-3. Ownership Manager Module Contract

ウォレットオーナー（基本的には単一の EOA）の管理を行う Module Contract。

EOA の場合、それに対応する秘密鍵を紛失してしまうと、そのアカウントが保有している資産を全て失うことになるが、コントラクトウォレットの場合、この Module Contract のようなコントラクトを用意し、所定の条件を満たした場合にウォレットオーナーを変更できるようにしておくことで、（ウォレットオーナーに相当する EOA に対応する）秘密鍵の紛失に対応することが可能である。なお、ウォレットオーナーを変更したとしても、ウォレット（Identity Proxy Contract）のアドレスは変わらない。

ウォレットオーナーを変更するための条件は様々考えられるが、昨今、その条件を「ウォレットオーナーがあらかじめ設定しておいたアカウント（ガーディアン）のうち半数以上の同意を得ること」とする、ソーシャルリカバリーと呼ばれる手法[1]（図 5）が有力視されており、unWallet もこの手法を採用している。

[1] https://vitalik.ca/general/2021/01/11/recovery.html

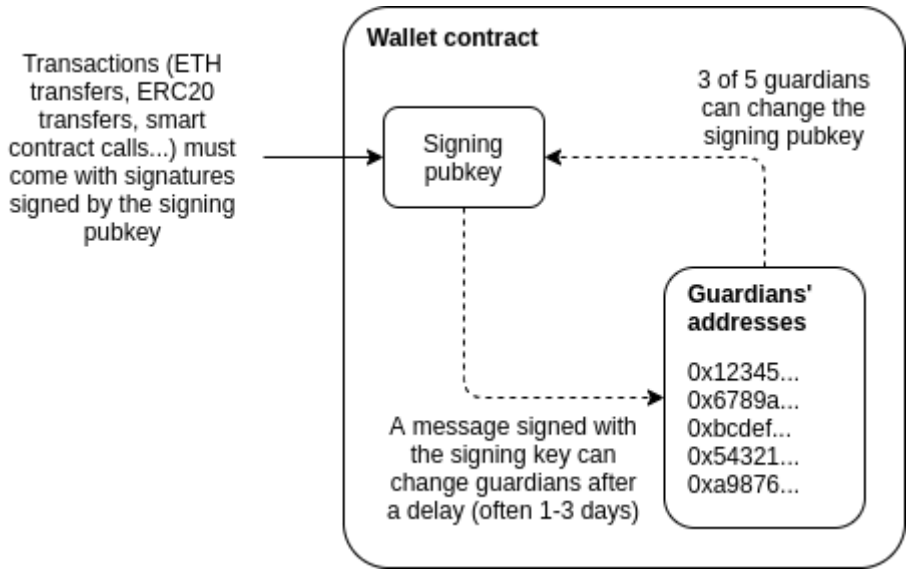

図 5. ソーシャルリカバリー
(https://vitalik.ca/general/2021/01/11/recovery.html より引用)

2-4. Transfer Manager Module Contract

FT（ERC20 トークン）や NFT（ERC721 トークンや ERC1155 トークン）の譲渡に対して制約をかけるためのモジュール。

昨今、著名プロジェクトの偽 Web サイトにユーザーを誘導し、そのユーザーが保有する FT や NFT を詐欺師アカウントに対して譲渡するようなトランザクションを実行させる、といった詐欺手法が流行しており、場合によっては数億円規模の被害が生じているが [2]、この Module Contract を利用して以下のような制約をかけることで、そのような詐欺に対抗することが可能となる。

- 自身が保有する FT や NFT をロックする（譲渡不可能な状態にする）
- 自身が保有する FT や NFT を譲渡可能なアカウントのホワイトリストを設定する
- 上記のような制約の申請から適用までに一定時間の経過を必須とする

なお、実際にこのような制約を機能させるためには、トークン譲渡に相当する外部コールを行う Relayer Module Contract との連携が必要である。すなわち、Relayer Module Contract に「外部コールを行う前にこの Module Contract によって設定された制約を確認し、条件に当てはまった場合は外部コールを行わないようにする」ような機構を実装する必要がある。

現在、高いリテラシーを有するユーザーであっても、リクエストされたトランザクションの内容を厳密に確認することは面倒かつ困難であるため、巧妙な詐欺を “注意する” だけで防ぐことは難しい。マスアダプションの進行に伴い、“万が一罠にかかってしまっても被害を受けない” を実現する上記のような機構は益々重要になるだろう。

[2] https://coinpost.jp/?p=344465