ANASTASIA LABS

Smart Handles - Final Milestone

Project Close-out Report

Project Number	1000011
Project manager	Jonathan Rodriguez
Date Started	2023, October
Date Completed	2024, October

ANASTASIA LABS

Contents

Smart Handles **1**

Introduction **2**

Objectives and Challenges **3**

Planning **4**

Recap - Smart Handles **5**

- On-Chain Stipulations 5
- Off-Chain Endpoints 5
- CLI Agent 6

Detailed list of KPIs and references **7**

- Challenge KPIs 7
- Project KPIs 7

Key achievements **8**

- Development of a Wrapper Contract 8
- Development of a Complete Off-Chain Suite 8
- Provision of a Sample Reference for Instance Developers 8

Key learnings **9**

Next steps **10**

Resources **10**

- On-Chain Contract 10
- Off-Chain SDK 10
- CLI Agent 10

Smart Handles

An abstract contract, enabling dedicated script addresses (enhanceable with ADA Handles) for interaction with DApps without going through their official frontends, and only with use of a wallet.

URL: Project Catalyst Proposal

Introduction

Considering how most DApps provide services through a single frontend, one can argue this centralized aspect can go against the ethos of the blockchain industry. While many of these user interfaces are open source, the inconvenience of going through having a local build makes it hard for excluded audience to use such DApps.

We offer an abstract smart contract, which developers can leverage to implement specific instances for dedicated tasks (such as swapping ADA to MIN via Minswap), lock an ADA Handle in the generated address, and allow wallet owners to simply send funds to the contract and benefit the underlying DApp.

Since at the time of proposal Cardano contracts where incapable of handling datum-less UTxOs, this project also includes providing configurations to open source wallets for supporting smart handles.

Objectives and Challenges

- Create an abstract contract as a foundation for implementing dedicated routing endeavors
- Accompany the contract with an off-chain SDK for developers
- Complement the suite with a CLI generator package so that smart handles instances can be utilized with much more ease
- Implement a simple (ADA-MIN swap via Minswap V2) and advanced (arbitrary swaps via MinswapV1) instances of smart handles as examples
- Submit sample transactions on preprod testnet
- Open a PR to an open source wallet, providing documentation for how users can benefit from the ADA-MIN smart handles instance

Planning

Our project comprised of five main phases:

Design

The wrapper contract needed to provide a few stipulations (detailed further down) in order to guarantee integrity for all instances. Furthermore, convenience for both users and router agents were of utmost importance.

On-Chain Development

As it is inevitable for most abstractions to introduce overheads, we aimed to minimize the performance cost by employing Plutarch for the wrapper contract.

Off-Chain SDK

Adhering to the same interface introduced in our other off-chain SDKs, we provided a consistent and robust interface between all endpoints. Due to the abstract nature of this contract, the SDK requires the users to provide functions with predefined structures.

CLI Agent

To facilitate a convenient experience for router agents, we also developed a CLI application generator function that allows instances to easily provide a simple CLI which works through standard input and accompanying JSON files.

Wallet Integration

Since smart handles instances can only work with datums, wallets needed minor configurations in order to provide the intended level of convenience for their users. By openning a PR in an open source wallet (GameChanger), we provided the basic guideline for wallets so that they could implement this feature if desired.

Recap - Smart Handles

On-Chain Stipulations

- The contract needed to provide incentive for decentralized agents to carry out requests of smart handles instances, therefore ensuring agents getting their fees was one of the requirements
- Instances needed to be parameterized by their destination addresses so that the contract could guarantee their proper continuations without fault
- Two variants must have been supported (which we call "targets"): single, for instances that only supported single routes per transactions, and batch, for contracts that utilized a staking script to support multiple routes per transactions with minimal overhead
- Users needed to be able to cancel their requests at will
- As some DApp instances could benefit from other values provided in a transaction, the framework needed to also provide an advanced interface on top of the simple case (which only housed a users' addresses)

Off-Chain Endpoints

All endpoints comprise of both single and batch target, while also supporting simple and advanced cases.

Request

The simple variant only requires specifying assets to be locked at the instance. The advanced variant on the other hand, is much more involved, requiring a multitude of specs:

- Whether an owner is associated with the request
- How much is the router fee for routing to the destination address
- Router fee for invoking the reclaim endpoint
- Required mint for the routing scenario
- Required mint for the reclaim scenario
- An additional callback for tweaking the transaction as desired

Reclaim

Similar to the request endpoint, only the advanced datum requires considerable configuration:

- An output datum maker function, which provides the users with inputs assets and all the fields stored in the advanced datum
- A set of configurations for handling a required mint for reclaim
- An additional callback for tweaking the transaction as desired

Route

Unlike other endpoints, the interface here offers both simple and advanced cases with various configurations:

- Both require datum maker functions, the only difference between the two is the input datum provided (simple for one, advanced for the other)
- The advanced case expects the required mint configurations for routing
- Both take an additional callback for tweaking the transaction as desired

CLI Agent

Given a `Config` value (a datatype defined in the package), users can generate a CLI application with three endpoints:

`monitor`

The primary endpoint for router agents, polling an instance's address periodically to route upcoming requests and accruing the fees. With a `--reclaim` flag the endpoint aims to invoke the advanced reclaim logic of the instance.

`submit-simple`

This endpoint will only require the Lovelace count to be provided, along with any additional assets to be locked at the instance.

`submit-advanced`

In addition to the Lovelace and assets arguments, this endpoint takes additional values to correspond directly to the off-chain SDK's advanced request endpoint.

Detailed list of KPIs and references

Challenge KPIs

Simplicity

- Simple datum for easier adoption by wallets

Flexibility

- Advanced datum for more elaborate instances

Tooling

- Off-chain SDK with consistent interface between endpoints
- CLI generator package for convenient product development

Project KPIs

All-Round Solution

- From on-chain wrapper contract to a CLI application that end-users can enjoy

Correctness via Tests

- Implemented off-chain tests using an emulator

Fully Documented for Easier Adoption

- Complete example of utilizing this solution

Key achievements

Development of a Wrapper Contract

A customizable contract that relieves adopters from certain validations and attack vectors such as agent reimbursement and double satisfaction.

Development of a Complete Off-Chain Suite

Accompanied by an off-chain SDK and CLI application generator, instance developers can easily achieve sample transactions, and consequently shipping a working product.

Provision of a Sample Reference for Instance Developers

The example provided within the off-chain SDK package provides an all-round use-case for the contract and the accompanied off-chain interfaces:

- **Minswap V1 & V2 Examples**

Key learnings

Ups and Downs of Adding Features to Frameworks

Ensuring flexibility in such a product proved to be a rewarding challenge, as support for each new feature propagated throughout the whole suite.

Importance of Emulated Tests

Without sample transactions carried out within an emulator environment, dealing with testnets would be much more time consuming.

Importance of Testnet Transactions

As much as emulated tests provide convenience, they won't replace real transactions submitted on chain, approvable by anyone.

Next steps

Support for Datum-less UTxOs

With activation of CIP-69, we can now avoid the required configuration by the wallets, and support direct deposits without any maintenance. This will offer the true vision of this product where users can interact with developed instances from any wallet and/or off-chain frameworks.

Resources

On-Chain Contract

- GitHub Repository

Off-Chain SDK

- GitHub Repository

CLI Agent

- GitHub Repository