ANASTASIA LABS

Smart Handles

Project Closeout Video Transcript

Project Number	1000011
Project manager	Jonathan Rodriguez
Date Started	2023-10-08
Date Completed	2024-10-07

Contents

Smart Handles

Slide 1

Hello Cardano, this is Mladen from Anastasia Labs, and today I'd like to present our closeout report for Smart Handles, which we initially called Smart Beacons.

Slide 2

OK let's start by getting a better understanding of what Smart Handles is.

In a nutshell, Smart Handles is a framework for generating contracts that users will be able to send funds to, in order to interact with the underlying DApp.

This enhances decentralization as it enables users to not be constrained by the websites of these DApps.

Slide 3

The framework itself makes minimal constrains, while also providing enough flexibility so that developers can have as much freedom as possible to implement instances for their users.

A common and useful example of this, is the idea of swapping tokens by simply sending some ADA to an address via a wallet, rather than using a DEX's interface.

Since each instance of Smart Handles will have a unique addresses, these addresses can be enhanced by locking ADA Handles in them, which will increase the convenience for end users.

Our framework is also a full-fledged suite: from on-chain framework, to a CLI application generator. This will give developers all the tools they needs to deploy their instances quickly and efficiently.

Finally, we've also opened a PR to GameChanger wallet, which can be used as a guideline for other wallet developers to support interactions with Smart Handles instances.

Slide 4

Now let's take a look at what one of these instance can look like in practice.

Slide 5

Imagine we have a swap Smart Handles instance for ADA to MIN via Minswap, and we store `$ada-to-min` ADA Handle in its address.

Users then can send some ADA to this handle, knowing 1 ADA will be sent to the routing agent.

A routing agent that's monitoring the address picks up the swap request and sends the funds to the Minswap address.

Finally, Minswap batchers handle the order and the original user gets MIN at market price.

You can see that this only required the user's wallet to support Smart Handles, and there was no need for the user to open up Minswap's platform.

Slide 6

In order to deliver this product, we had to meet some concrete goals.

Slide 7

First and foremost, the contract needed to be abstract enough to offer a reasonable flexibility for developers.

Of course the contract needed to be accompanied by an off-chain package.

On top of that, we also wanted further convenience, specifically for the routing agents, so that they could support this initiative more easily.

Slide 8

To simplify the learning curve for developers, we also wanted to have extensive examples to be used as references.

Having sample transactions on testnet was also something we considered a reliable proof of accomplishment.

Finally, a concrete guide for wallet developers to support Smart Handles was essential.

Slide 9

So we divided the road map into 5 phases.

Slide 10

First, designing the protocol and the overal interface provided by the suite. This gave us a clear path for development.

Slide 11

Developing the on-chain contracts was the second step. Knowing it's generally inevitable for abstractions to introduce overheads, we employed Plutarch to minimize this performance cost.

Slide 12

Following our other off-chain SDKs, we used the same interface for all the endpoints. This resulted in a consistent API, not only between all the endpoints of this project, but also accross our other off-chain SDKs.

Slide 13

In order to easily reuse the infrastructure laid out in our off-chain SDK, we opted to implement the CLI application in Typescript.

The package we implemented, requires the users to provide a set of configuration values and functions, and it'll generate a CLI application for their instances with a robust and familiar argument interface.

Slide 14

We found GameChanger wallet to be the most flexible open-source wallet in order to open a PR for integration.

Slide 15

Now let's take a closer look at each of the components.

Slide 16

First the on-chain contract.

Slide 17

First two primary validations are agent imbursement, and guaranteed route destination.

For simple Smart Handles, we hardcoded 1 ADA as the fee. But advanced instances allow custom fees for each request UTxO.

The route destination is also specified as a parameter. This ensures its correctness for instances.

We wrote the contracts for two "targets:" single and batch. Single target means only one route can be performed per transaction, while batch allows routing of multiple requests in one transaction.

There is also a chance for requests to get stuck at an instance. So we also implemented a reclaim mechanism. This is quite simple for the... well the simple case! It only requires the signature of its owner.

However, it allows for a much more involved validation for advanced reclaims, which we won't delve into here.

Similarly, routing of advanced datums also provides instances' logics with much more data and flexibility. We'll briefly take a deeper look later.

Slide 18

Let's see all the off-chain endpoints for interacting with a Smart Handles instance.

All endpoints support both single and batch targets, and also simple and advanced datums.

Slide 19

We refer to sending funds to a Smart Handles instance as "requests," since they are "route requests" to be processed by routing agents.

Common for all variants, some ADA and assets have to be specified to get locked. However, for the advanced datum, a few more values must be provided:

- The owner here is optional. Since advanced reclaims are primarily meant to be carried out by router agents, we needed to allow instances the option of preventing reclaims. So if no owner is specified, the request won't be reclaimable.
- As mentioned earlier, each advanced datum can have a custom router fee.

- To allow adjustment of incentives, it was reasonable to offer a separate field for the rewards agents could accrue when invoking the reclaim endpoint of advanced requests. So that's why we also have a "reclaim router fee."
- Two mint configurations, one for routing and the other for advanced reclaims.
- Finally, advanced datums offer a field for providing arbitrary data.

Slide 20

Since routes, whether simple or advanced, are primarily meant to be produced at another script address, both configurations need to specify how the output datum must be built.

Additionally, if the advanced datum has any mint requirements, they too need to be provided.

Both simple and advanced datums also allow any additional tweaks to the transaction.

Slide 21

For reclaiming of simple requests, no values are required to be provided. Because only owners themselves can reclaim their requests.

For the advanced case however, it is very similar to routing. So the configuration must specify:

- How the output datum must be built,
- If there are any mint requirements, they too should be specified,
- Any additional tweaks to the transaction can also be provided.

Slide 22

Now let's take a look at the CLI endpoints.

Slide 23

`monitor` is the primary endpoint for agent. It'll keep polling the instance's address, looking for requests to route.

Whenever it finds one, it tries to perform the route and grab its fee.

This endpoint also offers an optional flag for switching to advanced reclaims.

Slide 24

As its name suggest, `submit-simple` is for producing simple requests at the instance's address. It only requires the amount of Lovelaces that should be locked, and optionally any additional assets.

Slide 25

For `submit-advanced`, on top of funds to lock, advanced datum fields must also be specified. The generated application supports provision of additional information via an extra JSON file.

Slide 26

So we can summarize what we built in 3 main components.

Slide 27

- The on-chain framework contract
- A complete off-chain suite, both the SDK and its corresponding CLI generator
- Simple and advanced examples to server as a reference for easier familiarity with the suite

Slide 28

This was a relatively involved project and took us a while to reach the finish line. But we learnt a lot along the way.

Slide 29

First and foremost, we gained a better understanding of what it means to support a wide range of flexibility in a framework. While this gives much more freedom for instance developers, it also meant that for each feature we added, we had to propagate its support throughout all the components of our software suite.

Consequently, tests, both emulated and on-chain, proved of utmost importance. They were our firt line of assurance whenever we introduced a new feature.

Slide 30

At the time of our proposal, Cardano contracts were incapable of spending UTxOs that had no datums attached to them. But now, with Plutus V3, this is no longer the case and makes the main vision of this solution attainable. Meaning, users will be able to send funds to deployed Smart Handles without any configurations required, that is, using any wallet whatsoever.

Slide 31

These are the 3 repositories of Smart Handles, please consider giving them starts!

And with that, I thank you for your time, and wish you a great day!