

hydro

**Hydro Snowflake ERC-1484
Marketplace Coupon
Bounty
v1.0.0**

06.11.19



Luiserebii

<https://serebii.io>
<https://github.com/Luiserebii>

Table of Contents

1. Introduction 3

2. Brief Architecture 4

3. Deployment 6

4. Usage 8

5. Final Thoughts..... 9

1. Introduction

This project aims to accomplish the creation of a smart contract marketplace which can be deployed by any one entity, from businesses to individuals. This marketplace includes item tagging, delivery method specifications, the ability to toggle items from active to inactive (and vice versa), and coupon functionality.

Items and coupons are represented as ERC-721 tokens, as non-fungible symbols which can be transferred and owned. Upon creation, items are typically registered to the EIN of the address which sent the transaction, often being the seller. On purchase of the item, the ownership over the item is transferred from the seller to the buyer. Similarly, coupons are initialized as ERC-721 tokens, but are usually registered to the user intended to use it (i.e. potential buyers). Upon redemption, coupons are burned, thus enforcing their one-off usage.

The GitHub repo for the smart contracts can be found at:

<https://github.com/Luiserebii/HydroSnowflake-Marketplace-Coupons>

The GitHub repo for the Snowflake Dashboard integration piece can be found at:

<https://github.com/Luiserebii/HydroSnowflake-Marketplace-Dashboard>

2. Brief Architecture

In order to enforce separation of concerns, efforts to separate logic and functionality between contracts and “objects” have been made. As an example, the **Marketplace** contract holds the generic logic that a contract marketplace should have, as internal functions. **SnowflakeEINMarketplace** then builds on top of the **Marketplace** contract, exposing these functions, giving the contract a way to specify restrictions on function calls to EIN owners. Finally, our **CouponMarketplaceResolver** handles the actual logic behind the Snowflake Resolver in calling the Via contract.

Some additional refactors have been made for the sake of object-oriented principles. **SnowflakeReader** is a contract that shares the ability to easily read the Snowflake **IdentityRegistry** through a single function. All that is needed is to set the address of the **Snowflake** contract.

In order to take advantage of the unique methods in which we are determining identity (via EINs, instead of addresses), much of OpenZeppelin’s contracts pertaining to contract ownership and ERC-721s have been rewritten within the context of Snowflake.

A diagram on the next page elaborates on the smart contract architecture. Note that this is an older version, yet still preserves the same organizational idea.

This diagram can also be found at:

<https://camo.githubusercontent.com/8ce82d8a0317b15410a4d801a6514276555bc1cb/68747470733a2f2f736572656269692e696f2f687964726f2f696d672f536e6f77666c616b65253230436f75706f6e2532304d61726b6574706c6163652532302d253230487964726f253230426f756e74792e706e67>

Notes on differences from this diagram can be found here:

https://github.com/Luiserebii/HydroSnowflake-Marketplace-Coupons/blob/master/docs/diagram_change_notes

3. Deployment

The Hydro Snowflake Coupon Marketplace currently relies on an npm module called **solidity-deploy**. Although Truffle deployment scripts are available, much difficulty has been found in trying to validate the contracts on Etherscan. After weeks of frustration, I started to think that the Truffle migrate tool was creating some sort of bytecode corresponding to flattened code I could not determine. The contracts compiled by Truffle were often hundreds of bytes larger than contracts compiled by use of a flattening tool.

Since the Hydro Snowflake Dashboard requires verification of the Resolver contract, this quickly becomes an issue given the complexity of the deployment process. Manual flattening and compiling/deploying on Remix is possible, but does not scale. To overcome this issue, I spent the time writing a module to simplify the flattening and deployment process, which are used by the deployment scripts.

To deploy the Snowflake Coupon Marketplace, **cd** into the **/deployments** directory and run the following commands, replacing the empty values (signified by **[VAL]**) with their respective addresses. You will need to locate the address of the respective deployed contracts as needed.

Due to what is likely a web3 bug, larger contract deployments will run, but will not return to signal success. It is recommended to check the deployment address on Etherscan for new deployments, and to stop the script once the deployment is confirmed. These commands have been marked by asterisks (*).

./deployment.sh

(INIT and ITEMFEATURE)

node main-deploy.js --stage 3

(COUPON_FEATURE)*

node main-deploy.js --stage 4

(COUPON_MARKETPLACE_VIA)*

node main-deploy.js --stage 5 --CouponMarketplaceViaAddress [VAL] --CouponFeatureAddress [VAL] --ItemFeatureAddress [VAL]

(COUPON_MARKETPLACE_RESOLVER)*

node main-deploy.js --stage 6 --CouponMarketplaceViaAddress [VAL]

--CouponMarketplaceResolverAddress [VAL]

(SET_1)

node main-deploy.js --stage 7 --CouponMarketplaceResolverAddress [VAL]
(COUPON_DISTRIBUTION)

node main-deploy.js --stage 8 --CouponMarketplaceResolverAddress [VAL]
--CouponDistributionAddress [VAL]
(FINISH)

Successful example logs can be found in the **/docs** directory. Here is the most recent one:

<https://github.com/Luiserebii/HydroSnowflake-Marketplace-Coupons/blob/master/docs/HydroSnowflake%20Manual%20Deployment%20Log%2006-07-19>

4. Usage

Using the Hydro Snowflake Coupon Marketplace contracts are fairly simple. The resolver inherits two variables from the **Marketplace** contract: **ItemFeatureAddress**, and **CouponFeatureAddress**. In order to add, update, or remove items or coupons, you will need to interact with the respective contract to do so. There is no UI for this at the moment, so the most user-friendly option would have to be through Etherscan.

The best descriptions on how to use the contract is through the contract tests, which has documentation explaining each step. Please see:

<https://github.com/Luiserebii/HydroSnowflake-Marketplace-Coupons/blob/master/test/marketplace/coupon-marketplace.js>

The React front-end is self-explanatory. Simply select an item, type in a coupon code if available, and click "Purchase" to purchase the item!

5. Final Thoughts

I hope this documentation has been useful, and above all, clear.

If you would like to contact me, please email me at: luis@serebii.io

Thank you!
~Luiserebii~