Evolving Hook Tokens:
Customizable and Revisable Financial Instruments that Enable Advanced Fundraising Capability and Control

Robert M.C. Forster*

August 25, 2024

Abstract

In this paper we introduce Evolving Hook Tokens. Evolving Hook Tokens (EHTs) are tokens that have the ability to implement and transition unique functionality to add advancements to traditional tokens we use today. Throughout the transfer process of a token, there are checkpoints to execute additional behaviors. For example, a hook may: call out to tax logic that executes a tax on certain calls, call to logic that will use taxes to instantly share tokens between all users, or call to a decentralized exchange to execute arbitrage before a user transaction to get the user a better price. These hooks can be any approved arbitrary code, and they have endless possibilities. EHTs give companies the ability to profit from trading, encourage certain trading activity, upgrade their capability throughout time, and overall take full advantage of markets that until now have been untamable.

1 Introduction

The first modern stock was introduced in the 17th century, and since then fundraising for companies has remained relatively unchanged. With the advent of blockchain technology, companies began making tokens which in many ways function similarly to stock. They're often used to raise funds, and users often trade tokens between each other.

The programmability of blockchain tokens have allowed them to have advanced functionality such as charging users taxes on trading or allowing them to be used for governance capabilities. The features for tokens, however, have not progressed much further than that.

Generally tokens still have the same downside as stock where they're sold once for a cash infusion, and after that are set free on the market. Companies have few ways to profit off of users trading their token, they have no way to incentivize certain trading, and once they launch a token it can never have its functionality changed.

2 Solution

2.1 Overview

The solution to this problem is Evolving Hook Tokens. While we have ideas for hooks that will be proposed in this paper, the core theory behind EHTs is that not only does no one know best how to solve all of the problems with current markets, but companies in each stage of their lifetime will have different goals in what they want their tokens to accomplish.

If a company is brand new they may want to profit off of initial excitement with plain taxes; the hype around the company may then die down and they want to incentivize users with a staking system that splits taxes between all stakers to reward long-term community members; their product may then become very successful and they want to focus on encouraging users to provide liquidity or lend out

*Dedicated to Gunnar J.F. Johnson

their stock so they add a hook that degrades holdings, if they're not being used productively. Each stage of every company has different requirements, and EHTs exist to allow them to tailor how their token is being traded, and how they can best benefit.

Over the coming years we plan for hundreds or thousands of hooks to be created and utilized by EHTs. We'll slowly begin to discover strategies that work best for each situation a company may find themselves in, and educate companies on how to tailor a token to their needs. Eventually, EHT management will be core to a company's strategy and massively influence their fundraising abilities.

2.2 Architecture

Evolving Hook Tokens start with a default ERC20 token contract. They have child contracts that keep both logic and storage for the hook capabilities that will be called at different steps of interaction with the token. For example, a transfer will have multiple checkpoints where hooks are checked and called. There is a checkpoint before anything occurs for global updates and tax checks, a checkpoint after the initial subtraction of balance, a checkpoint after the addition of balance, and a final checkpoint for clean-up. Each checkpoint can call an arbitrary number of hooks depending on what's currently implemented in the contract.

2.2.1 Fallback

Hooks may have functions that do not exist on a traditional token. An example of this may be a 'claim' function if rewards are being distributed that do not go directly to a user balance. Because of this, a token must be able to forward calls to hook functions if a function does not exist on the parent contract.

To do this we'll be using a setup similar to the Diamond Standard where hooks with individual functions will save their function signatures in the parent contract which can then be fetched on an unrecognised function call. Upon addition of the hook, a unique function signature will be saved to a mapping where the function signature corresponds to the child contract address. When an unrecognised function call occurs, the parent contract fallback will check if the function signature exists in their hook functions mapping, and forward the call accordingly.

2.2.2 Balances

Some hooks, such as ones where users are dripped rewards over time, may result in dynamic balances that constantly change regardless of what's currently stored in the balance mapping. A call to balanceOf on the default contract will also have a checkpoint that checks if any of these dynamically updating hooks are active. Only one of these hooks may be active at a time. Having multiple active at once, such as one that degrades tokens over time and one that shares taxes over time, can easily result in mismatched balances and bugs. This restriction may be eased over time, but for our initial versions it is not worth the risk.

Dynamically updating balance must also check for the last time a user has updated their balance, and ensure that, if a hook has been deprecated after the last user interaction, the user balance is first updated to its final stage before updating based on the current token configuration.

2.2.3 Transfers

Transfers will contain multiple checkpoints: at the beginning of a transfer to make sure any updates are made before balances are adjusted and to check for taxes that must be removed, after the first subtraction of balance in case another update needs to be at that points, directly after addition of balance in case an update needs to be made there, and a concluding checkpoint on success of the others.

These will likely contain tax logic, gamification logic, and logic for any other hook such as lending pools or updating lottery entries. At each checkpoint the parent contract will loop through a list of contracts that require updates at that point, call the corresponding checkpoint function on the child contract, and update variables in the function as needed.

2.2.4 Winding Down

Some functionality will not be able to be immediately stopped and evolved. For example, a Vault Hook (which uses Ether from taxes to build up a reserve that tokens may be redeemed for) may have built up Ether in its reserves that is not normally used in plain ERC20s. In this case a hook will need to be wound down before it can be closed or switched to something else.

This will generally happen by converting any hook-specific balances into traditional balances. For example, in the case of Vault hooks, Ether may need to be converted in bits at a time for tokens and those tokens then distributed to holders. At the end of the winding down period, this results in users only having balances that would be used in a traditional ERC20.

There may also not be much of a need for winding down, such as switching tax types. If you're switching from a plain tax to a time-based tax, the contract can simply send tokens from the plain tax child contract to the new time-based one and update the parent contract.

2.2.5 Evolution Delay

After winding down (or in some cases while winding down is happening), there will be a required delay before a full switch occurs. This delay will likely be in the realm of two days. It's required to allow time for users to cash out of a token if the owner is evolving the token in a way that they disagree with.

For example, if a token was initially started with a Taxshare hook and that's the primary reason a user is holding it, and the owner decides to remove that capability, they should not be forced to abide by the new hook before they have the opportunity to cash out.

Delays ensure that token owners cannot catch users off-guard with hook evolutions that may not be desired.

3 Example Hooks

There are an infinite number of hooks that can be implemented in EHTs. Functionality can range from adding taxes to creating DeFi protocols within the token itself. Although there are hooks that cannot be used together, most of them are designed to be able to be used in tandem. For example, tax types provide the dual benefit of disincentivizing behaviors through charges applied to trades while also adding funding that can be used to incentivize different or opposite behaviors with gamification. Tokens can have any number of additional functionalities and continue to add more as time goes on.

3.1 Taxes

3.1.1 Demurrage

Demurrage tokens charge users for holding tokens unproductively. It does this by degrading all balances a certain percent per year, unless the token is being held by a safe haven. These safe havens may be liquidity pools, lending protocols, staking contracts, or any other protocol a company wants to encourage depositing into. Economically, penalizing users for not using their tokens productively provides the same or better incentive as rewarding them for using their tokens productively, and it comes with the added benefit of being able to use degraded tokens however the company sees fit.

3.1.2 Trade-based Taxes

Trade-based taxes are the traditional taxes that take a percentage of every trade when a token is bought or sold. If $100 of tokens is purchased with a 2% tax, $98 of tokens are received.

3.1.3 Profit-based Taxes

Profit-based taxes would only tax trades based on the amount of profit a user made. If a user purchases $50 of tokens, the token price rises so that those become worth $100, then the user sells the token, a

2% profit-based tax will be taken from the $50 profit and $99 will be returned.

This is achieved by checking the price of the token every time a transfer occurs and updating a user's average cost basis when tokens are received. When selling, the average cost of tokens can then be compared to the current price and taxed accordingly.

3.1.4 Time-adjusted Taxes

Time-adjusted taxes can be used with either trade- or profit-based taxes. They start at a high amount then lower as time goes on. For example, it may be a 50% tax if you sell in the same block, then linearly lower to a 0% tax if you sell a month after buying. This incentivizes holding tokens longer, similar to short-term vs. long-term capital gains taxes charged by governments.

3.1.5 Volume-adjusted Taxes

Volume-adjusted taxes can also be used with either trade, or profit-based taxes. These will adjust taxes based on the amount of buy and sell volume occurring recently. While there are many ways this can be implemented, one method may be by calculating net buy or sell volume in the recent period and proportionally increasing and decreasing taxes on each side. This functionality uses the same incentives of normal price discovery (e.g. when people are selling despite no major underlying changes, buying becomes more attractive) but magnifies them to smooth out volatility.

3.2 Gamification

3.2.1 Vaults

Vault tokens will use taxes from trading to provide a lower bound of token price. Taxes will be traded for Ether, stored in the contract, and tokens can be burnt for their proportional amount of Ether in the contract. As more trading occurs, more Ether builds up in the contract, and the lower bound of token price will increase because of the backing funds.

3.2.2 Lottery

Lottery tokens could be an example of benefits that taxes can provide to encourage users to trade the token. A lottery token may entail a chance to win extra tokens on a purchase, or a chance to win tokens when holding your own. Benefits such as this can provide users extra reasons to buy or hold a token.

3.2.3 Taxshare

Taxshare tokens share taxes between all holders. When a trade is made, taxes are immediately proportionally distributed between all holders of the token. By sharing taxes, a team encourages users to continue holding the token through turbulent times.

3.2.4 Charity

Charity tokens use all taxes to donate to charities. Taxes will be immediately converted for Ether, and sent to a charity of the token creator's choice. These enable the trading of tokens that may otherwise not have much societal value to benefit worthy causes.

3.3 Additional Functionality

3.3.1 Lending

Lending protocols can be built directly into the token. The lending pool would be a child contract dedicated to this token that will be able to check for required liquidations any time a token is sold. It can then also automatically liquidate any positions necessary, ensuring the protocol never incurs bad debt.

3.3.2 Staking

Staking functionality could be used for myriad reasons including: governance purposes, to decide on who should receive benefits from taxes, or reasons pertaining to the purpose of the token itself. Staking can take many forms but will generally involve locking tokens up for a period of time and rewarding users for doing so.

3.3.3 Arbitrage

Before any decentralized exchange (dex) trade the token contract could have the ability to arbitrage between other dexes. If it detects prices are uneven between multiple dexes, the token can automatically trade before the user's transaction to even out prices, give users the best price possible, and make a profit while doing so.

3.3.4 Airdrop

Airdrop functionality allows teams to distribute Ether or tokens to holders. It can allow teams to distribute profits from their protocol proportionally equal to the amount of tokens users hold, or distribute any other type of token to holders.

3.3.5 Governance

Governance functionality is one that's already common in many tokens. Token balances can be queried by a governance contract to determine how many votes a user had at a certain time. The problem currently is that a token must be initially launched with governance capability, or it must otherwise use a wrapper to enable it.

4 Conclusion

Evolving Hook Tokens will enable the next generation of trading. There's a massive amount of potential for companies to both take advantage of and control trading with methods that are currently ignored, and the methods that aren't being ignored can never be adjusted based on a company's circumstance. Trading of tokens and stocks shouldn't just be a casino that investors bet on with no care for the underlying company. Tokens should be fully utilized as a way to benefit the company where it needs help at every step throughout its life. Whether that's receiving extra funding, encouraging patience, incentivizing productivity, or simply giving a purpose to trading further than money, there will always be ways in which a company can benefit from customizing its token to its needs.