ARC

OFFICIAL **WHITEPAPER**

ARC : A Decentralized Workflow Engine and Semantic GUI

Abstract. A decentralized, heterogeneous workflow engine with a semantic programmatic GUI would allow all network constituents and future peers within a network to build, manage, connect, and deploy their data and products, both front-end, and smart contracts intra-chain and to the web seamlessly, and much more efficiently than alternative methods. Existing blockchains provide part of the solution needed for the mass adoption of Web3 technologies. But lack significantly when it comes to cross-chain compatibility, creating exploitative opportunities for bad actors, slow iteration, deployment speed, and reliance on centralized services. ARC proposes a solution to these problems via a decentralized workflow engine. Making all input capabilities available through a semantic GUI, and all output processes via a decentralized backend.

1. Introduction

1.1 Web3 Development

The development of Web3 has traditionally been a slow and tedious process. It's plagued by communication problems between technical and non-technical team members. Moreover, web3 (not unlike web2) is littered with malicious and bad actors exploiting seemingly unavoidable security flaws and oversights. ARC's development ecosystem offers solutions for speed, cost, scalability, and security. ARC achieves this through our suite of tools that make product development and deployment easier and more secure, each in its own unique way.

Our development ecosystem not only streamlines the way web3 applications get developed but also has far-reaching implications on the way data gets managed across the web.

1.2 Data Management

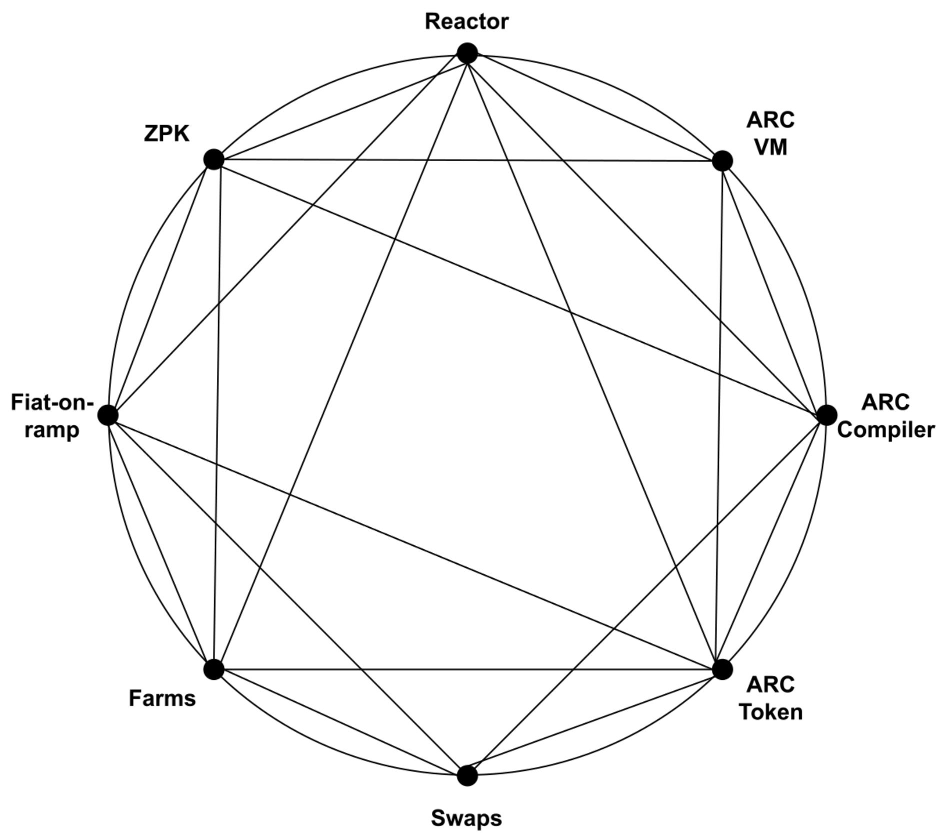
Transference of data on the internet has traditionally been controlled by centralized servers acting as custodians of user data. This system has worked well enough in the infancy of the internet, but in the age of information, it has its drawbacks. The potential for misuse of user data, difficulty in quantifying the data, cost of maintenance and set-up of the servers, and susceptibility to manipulation by bad actors, are just a few. Blockchains offer a strong alternative but are slower, less reliable than other technical counterparts, and still open to manipulation. The ARC Reactor addresses these issues through a decentralized workflow engine and semantic GUI for all Web3 data transference. This system serves as a decentralized liaison, onboarding, and automation system for all of Web3. The Reactor offers a solution for efficient and reliable data transference on the internet by allowing all network constituents to build, manage, connect, and deploy their data intra-chain and web seamlessly. The ARC Reactor operates without reliance on centralized actors or institutions.

User to Chain: The ARC Reactor allows for the seamless transference of data from peer to the blockchain. Users can easily build, connect, and deploy their data on any blockchain, utilizing the Reactor's UML mapping and OOP language to understand and manipulate the code. The Reactor also includes an auto-compilation feature, which standardizes uploaded smart contracts for easier deployment and iteration.

Chain to Chain: The ARC Reactor also facilitates the transfer of data from one blockchain to another. Users can easily build, connect, and deploy their data between different chains. The decentralized nature of the Reactor ensures that data transference is secure and equal among all network constituents, creating a self-sustaining system.

Chain to User: The ARC Reactor allows for the seamless transference of data from the blockchain to the user by utilizing existing methods of chain-to-user communications, paired with a flexible development environment. Users can easily build, connect, and deploy their data from the blockchain.

2. The ARC Ecosystem



2.1 ARC Reactor IDE

The ARC Reactor Integrated Development Environment (IDE) constitutes the core of our ecosystem. The Reactor IDE boasts a fully featured solidity IDE. While it allows for all standard functions of an IDE, the Reactor IDE is differentiated by way of a proprietary code-to-diagram technology that represents any smart contract into editable visual diagrams. This offers a lower barrier to entry for understanding how smart contracts function. The visual editor provides intricate control over every aspect of a smart contract, in effect creating a no-compromises, low-code solution to smart contract development.

2.2 ARC VM

The ARC Virtual Machine (VM) is a shared virtual CPU that can be run through the ARC Reactor IDE. It allows developers to test and run their smart contracts through the Reactor prior to deployment. The ARC VM differs from other VMs in that any contracts compiled to run on the ARC VM become intra-chain compatible. In effect the ARC VM allows developers to rapidly deploy any EVM chain code to any other EVM. Developers can also leverage the ARC VM to develop for any existing EVM chain. In order to compile smart contracts into byte code which gets executed on the ARC VM, the ARC VM utilizes the ARC Compiler.

2.3 ARC Compiler

The ARC Compiler allows for the streamlined deployment and iteration of smart contracts within the system. Upon uploading, the contracts are automatically compiled into a standardized format, ensuring that they are easily understood by the system and improving overall efficiency and reliability. In addition, ARC's automatic compile feature massively reduces the amount of tedious and redundant work developers are required to do to compile smart contracts.

2.4 ARC Language

In order to allow the Reactor IDE to map all functions and relationships of any smart contract into visual diagrams, the ARC Language was created. The ARC Language is a unique smart contract programming language that instantly maps all smart contracts across all EVM chains into Entity Relationship Diagrams (ERD), allowing users to edit the source code directly from activity diagrams. This powerful language is the driving force behind our GUI in the ARC Reactor IDE.

2.5 $ARC Token

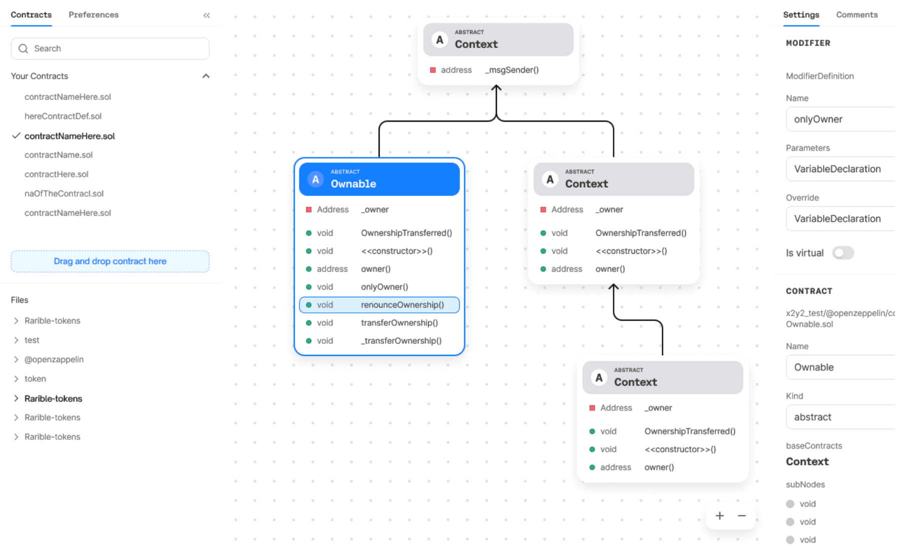
The $ARC Token is the heart of our decentralized system ensuring bad actors with malicious intentions are not incentivized to use the ARC Reactor. In addition to a monthly fee, The $ARC Token needs to be staked in order to access the ARC Reactor. By staking more $ARC, users can reduce the total amount of their monthly fees and unlock other benefits.

3. Reactor IDE & Semantic GUI

The semantic GUI of the ARC Reactor is a programmatic interface that allows users to easily build, connect, and deploy their data within the network. The GUI utilizes a unique OOP language that is organized into three layers:

3.1 ERD Diagrams

ERD, or Entity Relationship Diagrams, are a visual representation of the relationships between different entities within a database. In the context of the ARC Reactor, ERD diagrams are used to provide a visual representation of the relationships between smart contracts, their subnodes, and children. This allows users to easily understand the structure of their codebase and make changes as needed.[1]



MODIFIER

ModifierDefinition

Name: onlyOwner

Parameters: VariableDeclaration

Override: VariableDeclaration

Is virtual

CONTRACT

x2y2_test/@openzeppelin/c
Ownable.sol

Name: Ownable

Kind: abstract

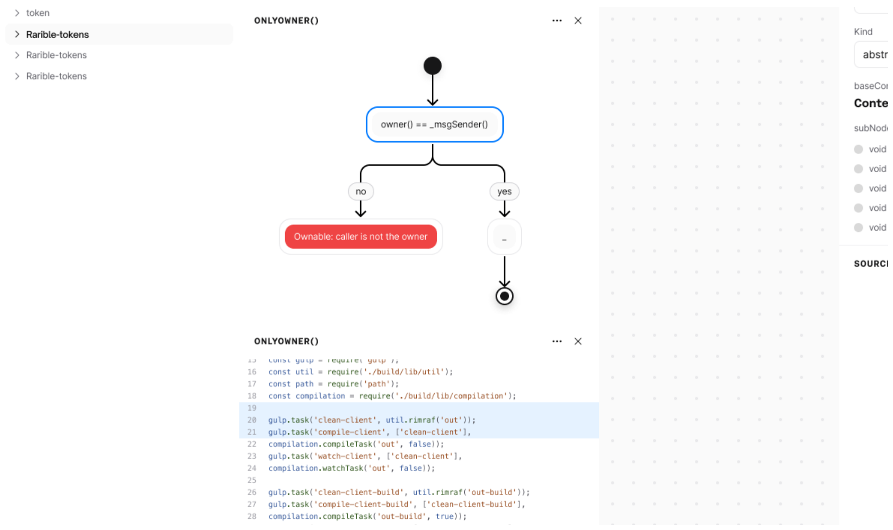
baseContracts: **Context**

subNodes
- void
- void
- void

3.2 Activity Diagrams

Activity diagrams are a type of flowchart that describe the flow of functions, actions, and events within a process or system. In the ARC Reactor, activity diagrams are used to describe the functions and interactions of the codebase, allowing users to easily understand the logic behind their contracts. These diagrams are particularly useful for users who may not have advanced programming skills, as they provide a visual representation of the code that is easy to understand.

[1] *Refer to the glossary on page 20.*



```
const gulp = require('gulp');
const util = require('./build/lib/util');
const path = require('path');
const compilation = require('./build/lib/compilation');

gulp.task('clean-client', util.rimraf('out'));
gulp.task('compile-client', ['clean-client'],
compilation.compileTask('out', false));
gulp.task('watch-client', ['clean-client'],
compilation.watchTask('out', false));

gulp.task('clean-client-build', util.rimraf('out-build'));
gulp.task('compile-client-build', ['clean-client-build'],
compilation.compileTask('out-build', true));
```

3.3 Interface and Libraries

The interface layer of the ARC Reactor's semantic GUI allows users to edit the code diagrammatically, or directly from the GUI. This layer also includes a variety of libraries for different data structures and input fields, allowing users to easily access the resources they need to build and deploy their contracts. The GUI also includes support for a variety of denominations and other features that make it easy for users to generate source code with just a few clicks. Overall, the semantic GUI of the ARC Reactor is designed to be user-friendly and intuitive, making it easy for users of all skill levels to build and deploy their contracts within the network.

4. Semantic Virtual Machine

The ARC VM is a core component of our system, providing a secure and efficient environment for running smart contracts and connecting cross-chain.

One of the integral offerings of the ARC VM is its speed. It is designed to be over 90% faster than other VMs on the market, allowing it to handle large volumes of transactions without slowing down the network. This makes it an ideal solution for high-throughput applications, such as decentralized finance (DeFi) and gaming. [2]

4.1 How the ARC VM achieves this impressive speed

There are a few key factors at play. First, the ARC VM uses advanced optimization techniques, such as just-in-time (JIT) compilation, to compile smart contracts as they are executed. This allows the VM to optimize the code for the specific input data and execution environment, resulting in faster and more efficient execution.

Second, the ARC VM uses a lightweight and flexible architecture that minimizes resource usage. This allows it to run on a wide range of devices and environments, from high-performance servers to low-power IoT devices. This flexibility enables the VM to operate efficiently across a wide range of applications, from DeFi to GameFi.

In addition to its speed, the ARC VM also offers a range of other benefits, including enhanced security and interoperability with other EVM chains. This makes it an essential part of our technology stack, enabling users to build faster, more securely, and in an interoperable blockchain ecosystem.

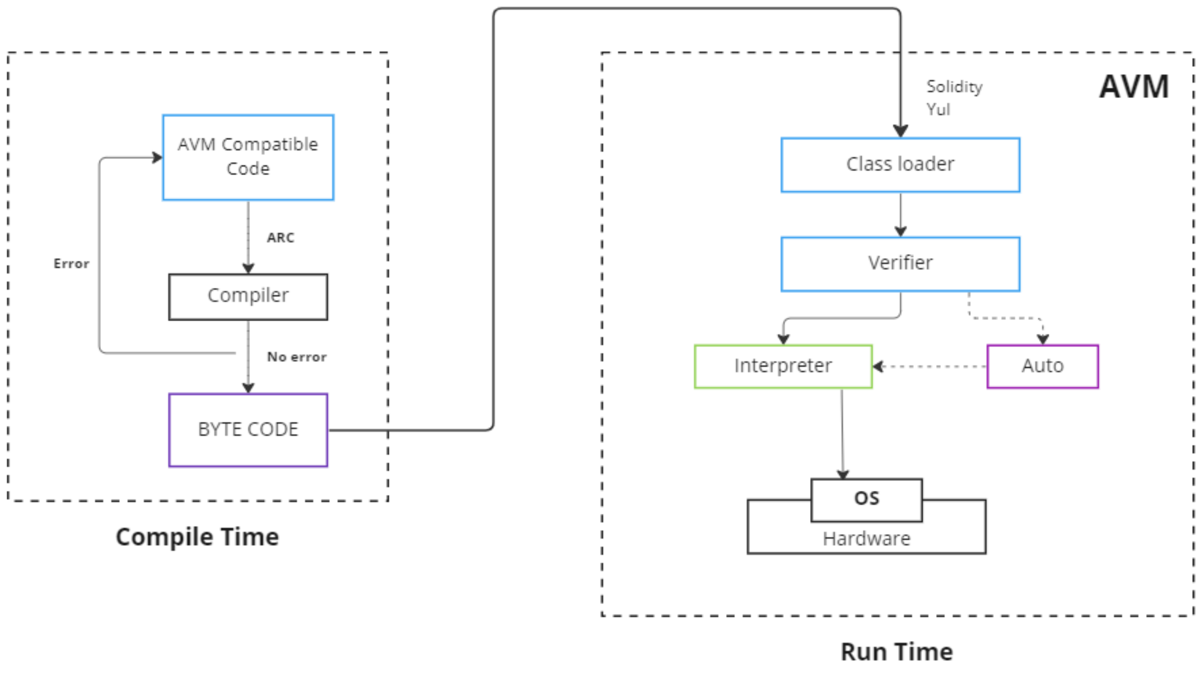
[2] *Refer to the glossary on page 20.*

5. Auto-Compilation - Automated compilation process for EVM

The ARC Reactor's auto-compilation feature allows for the streamlined deployment and iteration of smart contracts within the system. Upon uploading, the contracts are automatically compiled into a standardized format, ensuring that they are easily understood by the system and improving overall efficiency and reliability. This feature is particularly useful for users who may not have advanced programming skills, as it simplifies the process of building and deploying contracts within the network. Overall, the auto-compilation feature is a key component of the ARC Reactor's design, as it streamlines the process of building and deploying contracts within a decentralized network.[3]

5.1 Benefits of Auto-Compilation:

(1) Increased efficiency - users no longer need to manually compile contracts before uploading, saving time and resources; (2) Improved reliability - standardization of contracts ensures that they are easily understood by the system, reducing the risk of errors or miscommunication; (3) Greater accessibility - simplifies the process of building and deploying contracts for users with less programming experience.



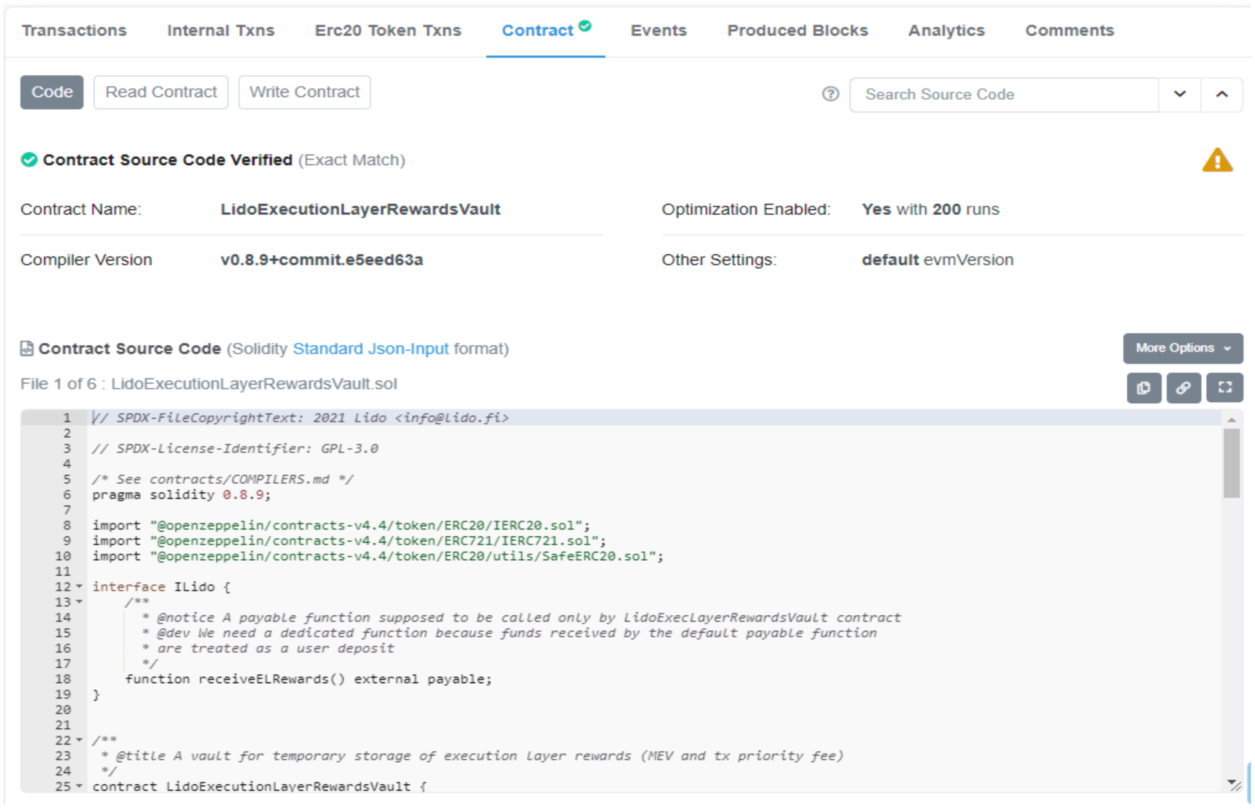
---

[3] *Refer to the glossary on page 20.*

6. Decentralized GUI

Through a decentralized GUI, ARC is reimagining what a block explorer looks like. Commonly, block explorers are used for mapping out certain transactions within a blockchain, as well as allowing users to delve into the smart contract that the transaction(s) interacted with. The main problem is that delving into the code is time intensive and has to be done line-by-line. This problem is attenuated by decentralizing the semantic GUI outputs, so every actor within Web3 can view code in a transparent manner, and understand how the logic works perfectly.

Note that the code for this smart contract is hosted on-chain and has to be examined line-by-line in order to understand the underlying logic. A large barrier for all users, especially the non-technical.



Contract Source Code Verified (Exact Match)

Contract Name:	**LidoExecutionLayerRewardsVault**	Optimization Enabled:	**Yes** with **200** runs
Compiler Version	**v0.8.9+commit.e5eed63a**	Other Settings:	**default** evmVersion

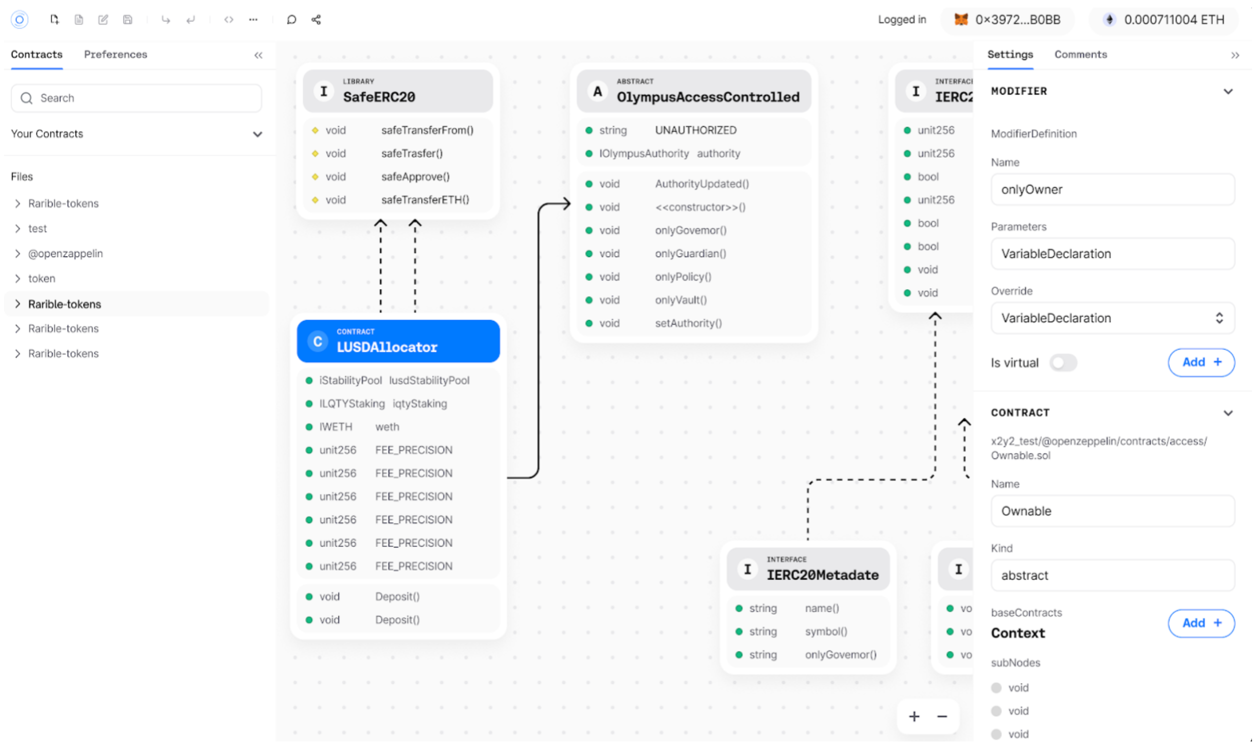
Contract Source Code (Solidity Standard Json-Input format)

File 1 of 6 : LidoExecutionLayerRewardsVault.sol

```
// SPDX-FileCopyrightText: 2021 Lido <info@lido.fi>

// SPDX-License-Identifier: GPL-3.0

/* See contracts/COMPILERS.md */
pragma solidity 0.8.9;

import "@openzeppelin/contracts-v4.4/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts-v4.4/token/ERC721/IERC721.sol";
import "@openzeppelin/contracts-v4.4/token/ERC20/utils/SafeERC20.sol";

interface ILido {
    /**
     * @notice A payable function supposed to be called only by LidoExecLayerRewardsVault contract
     * @dev We need a dedicated function because funds received by the default payable function
     * are treated as a user deposit
     */
    function receiveELRewards() external payable;
}


/**
 * @title A vault for temporary storage of execution layer rewards (MEV and tx priority fee)
 */
contract LidoExecutionLayerRewardsVault {
```

ARC's solution - ***A Decentralized Semantic GUI-based explorer for all chains***

With an accessible GUI interface for exploring smart contracts on any EVM chain, users can easily navigate through the code via an internet browser. Understanding functions from an entity relationship point of view, as well as activity.



MODIFIER

ModifierDefinition

Name: onlyOwner

Parameters: VariableDeclaration

Override: VariableDeclaration

Is virtual — Add +

CONTRACT

x2y2_test/@openzeppelin/contracts/access/Ownable.sol

Name: Ownable

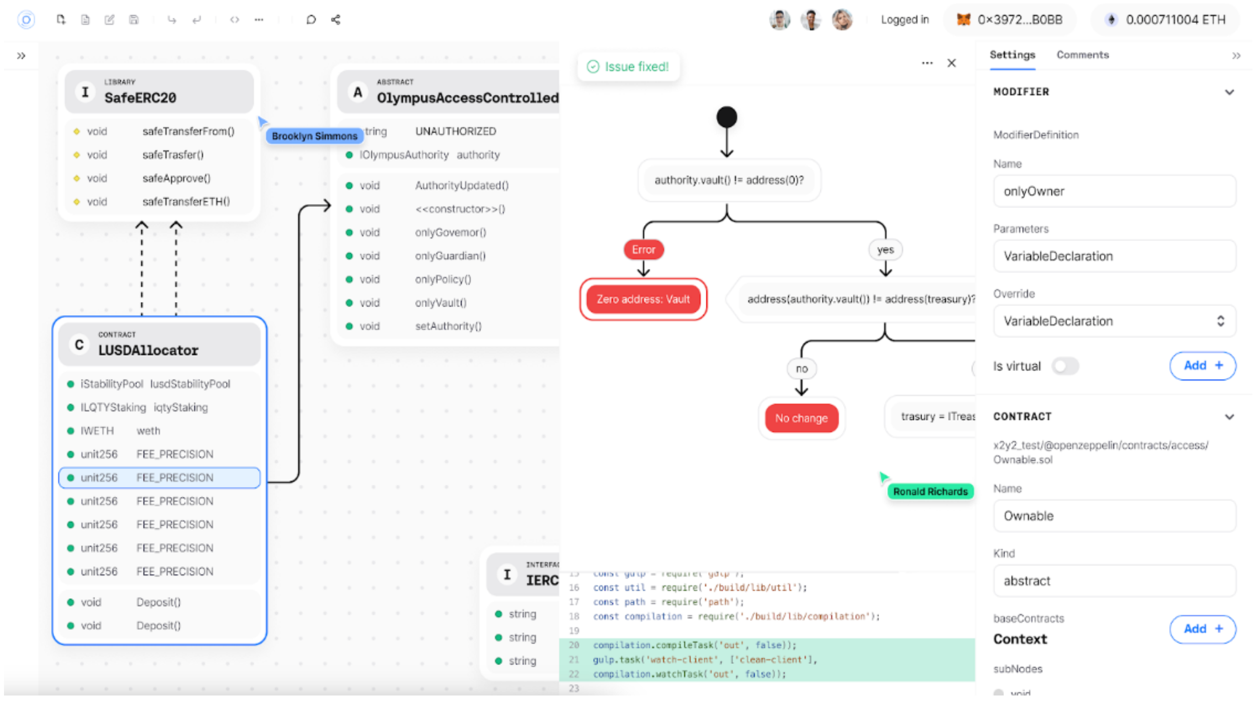
Kind: abstract

baseContracts: Context — Add +

subNodes: void, void, void

6.1 On-Chain Smart Contract Management

Utilizing the block explorer, teams will be able to upload deployed code into the reactor, via exporting. Or by logging into a wallet that can access the smart contract. One can edit directly from the block explorer.



```
const gulp = require('gulp');
const util = require('./build/lib/util');
const path = require('path');
const compilation = require('./build/lib/compilation');

compilation.compileTask('out', false));
gulp.task('watch-client', ['clean-client'],
compilation.watchTask('out', false));
```

MODIFIER

ModifierDefinition

Name: onlyOwner

Parameters: VariableDeclaration

Override: VariableDeclaration

Is virtual — Add +

CONTRACT

x2y2_test/@openzeppelin/contracts/access/Ownable.sol

Name: Ownable

Kind: abstract

baseContracts: Context — Add +

subNodes

7. ARC Token

The $ARC Token is the heart of our decentralized system ensuring bad actors with malicious intentions are not incentivised to use the ARC Reactor. In addition to a monthly fee, The $ARC Token needs to be staked in order to access the ARC Reactor. By staking more $ARC, users can reduce the total amount of their monthly fees.

7.1 The virtues of the $ARC Token:

- Stake $ARC to access Reactor (and disincentivize bad actors) via slashing. Securing the future of a safer WEB3 for all.
- Savings (1): SaaS - priced in stables / FIAT. Monthly packages that scale per user, features, and support. ARC savings
- Savings (2): Push Fees - Paid in $ETH. Users can save on gas fees dynamically based on the amount of $ARC they stake
- Revenue sharing on the future marketplace for Dapps and Reactor templates created by users

By using the ARC token as a barrier to access the ARC Reactor an added layer of control over what the ARC Reactor is being used for was developed. As it's a very powerful tool that could be maliciously used to rapidly generate clones of projects, anchoring the transactions in $ARC is essential for an added layer of security. In addition, this allows our supporters to invest in our developing ecosystem.

Example of fee savings from staked $ARC

Amount of ARC Staked	Savings per month
10,000	*0%*
25,000	*5%*
75,000	*15%*

7.2 What is $stARC?

$stARC (Staked $ARC) allows $ARC holders to stake their tokens in return for a synthetic token called $stARC. With this structure, users accumulating and staking $ARC will earn and be able to sell $stARC tokens accessing the $stARC liquidity pool.

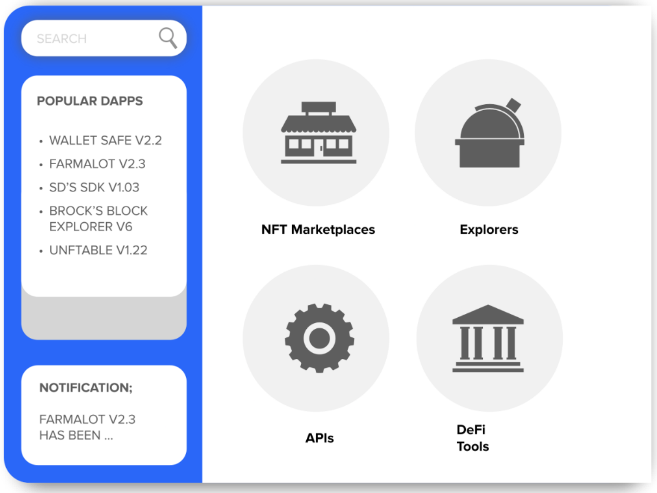
7.3 The virtues of the $stARC Token

Holders of $ARC are incentivized to accumulate and hold the $ARC utility token and become part of an ever-growing symbiotic ecosystem.

1. *$ARC holder selects contract*
2. *Contract locked in, $stARC (a synthetic token) is generated*
3. *The user holds $ARC in their wallet until the contract is complete*
4. *The user exits the $stARC contract via selling, accessing liquidity/ revenue via LP*

8. Decentralized App Store

The decentralized app store is a platform that allows users to discover and access decentralized applications (dApps) on the blockchain, as well as provides a place for developers to host the decentralized applications they make with the Reactor. The decentralized app store offers several benefits to developers and users alike.[4]



Early conceptual mockup of ARC's dApp Store

8.1 For Developers

The decentralized app store provides a secure and decentralized platform to host their dApps. This means that they do not need to rely on a centralized approval process or worry about censorship, and can be confident that their dApps will be accessible to all network constituents. In addition, the decentralized app store allows developers to easily discover and access dApps created by other developers, fostering collaboration and innovation within the ecosystem. dApps are not required to share a significant portion of their revenue, nor be constrained to one device. Because the reactor is heterogeneous and works across all chains, mobile, and the web, so will the dApp store.

8.2 For Users

The decentralized app store offers users greater control over their data and privacy. Unlike traditional app stores, dApps do not have access to personal information and users can be confident that their data is secure. In addition, the decentralized app store allows users to discover and access a wide range of dApps. Perhaps most importantly, the dApp store will feature the most transparency and security on the market. As every dApp is made with the Reactor and hosted in-store, while the smart contracts are hosted on our decentralized GUI, users can be sure of the exact functions the dApp consists of, what it operates on, and the PoR. Users will also be notified of any changes to any dApp codebase that is being used in their project.

[4] *Refer to the glossary on page 20.*

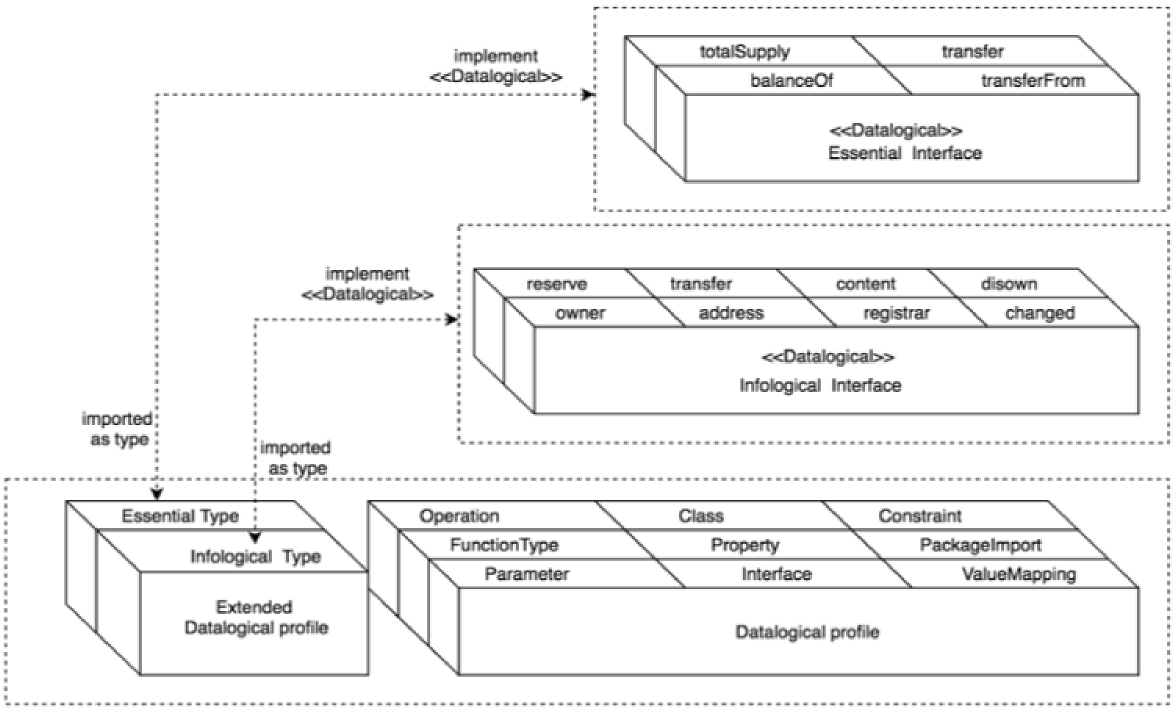
9. Heterogenous

The ARC Reactor is designed to be heterogeneous, meaning that it is able to work with a variety of different networks and protocols. This is achieved through the integration of multiple programming languages such as Solidity, Yul, and Bytecode. This allows the Reactor to connect and integrate with all EVM chains and most devices, enabling users to access, manage, deploy, and edit their data or smart contracts from any location; with or without an internet connection. The versatility and flexibility of the ARC Reactor make it a revolutionary technology for the web.

The heterogeneous nature of the ARC Reactor allows for seamless integration with a variety of networks and protocols, including:

(1) Solidity, the most popular programming language for Ethereum smart contracts; (2) Yul, a low-level programming language used in Ethereum; and (3) Bytecode, a compiled code that can be executed by a virtual machine.

This wide range of compatibility allows users to connect, manage, and edit their data or smart contracts from any location, with or without an internet connection. Essentially creating a bridge between the blockchain, web, and user that didn't exist previously.

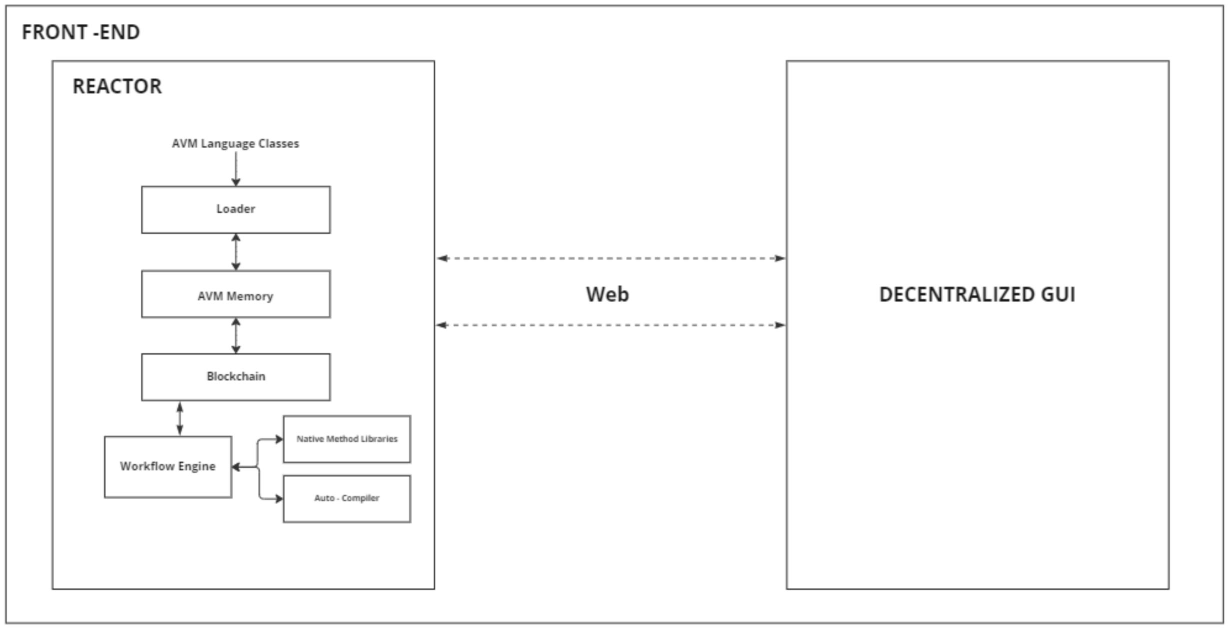


10. Decentralized - Requiring No Backend or Internet Connection

The decentralized workflow engine of the ARC Reactor allows for the seamless management, execution, and deployment of workflows within all EVM blockchains without a hosted backend, or internet connection for all processes besides deploying to mainnet. This decentralized nature ensures that the system is not reliant on any single point of failure, making it more resilient and secure.

One of the key benefits of the ARC Reactor's workflow engine is its lightweight design. This makes it easy for anyone to upload and test contracts locally via the ARC Virtual Machine within the engine. The ARC Virtual Machine is lightweight and compatible with all EVM chains, built with our proprietary language. Allowing users to test and deploy their contracts without the need for an internet connection or hosted backend. This enables users to easily test and debug their contracts, leading to faster deployment and iteration.

No backend is required. The reactor itself does not require a backend and all data is deployed directly to the front end and on-chain. The ARC Reactor has been designed to run completely in-browser, meaning all of the processes and functions needed to create Diagrams, run the Virtual Machine, or Compile the contracts are stored on the user's computer. The ARC Reactor utilizes local processing power to securely compute changes to smart contracts before committing to a mainnet.



10.1 Secure In-Browser Local Processing

Traditionally, when an individual visits a webpage, the browser will download the necessary files to display the webpage from a server. Once the webpage is downloaded, users will have access to all the information comprising that webpage on their local machine. This has some serious implications for how and where processing can occur.

Using proprietary methods, the ARC Reactor’s source code can be hidden and executed within the browser making it impossible to be accessed or tampered with. Because of this, users can interact with the Reactor without the need for a back-end or internet connection. This allows for much greater flexibility. This also implies that if in the event the ARC Reactor becomes infected with malware, this will not impact the experience for any other ARC Reactor users. The maximum impact of any such security breach will be localized to the infected machine.

10.2 Front-end, Server, Database

Traditional applications that communicate with both the user and the chain require some type of backend service to regulate communication between the server and the database. Queries get made by the user and sent to the server, which then makes a call to the database.

10.3 Front-end, Chain

Using ARC's secure local processing the backend service becomes obsolete as everything can be done on the front-end which the user interfaces with. By doing all this on the front end, users can now store data directly on chain eliminating the need for a backend or traditional database. Using this methodology, the blockchain in effect becomes the data storage point for applications. This offers a more secure and cost effective way for developers to create user-facing applications.

10.4 Impact on Web3

The decentralized nature of the ARC Reactor and it's programmatic capabilities allow it to scale Web3 not just for developers but for end-users, and business owners alike. Currently, well over a trillion dollars is spent in infrastructure worldwide for centralized servers and several billion per annum in energy expenditure. That could be attenuated by the workflow engine and semantic GUI ARC offers.

CONCLUSION

The ARC Reactor is a decentralized, heterogenous workflow engine with a semantic programmatic GUI that allows for the efficient management, execution, and deployment of workflows across all EVM blockchains, added languages, and the Web. Its decentralized nature ensures accessibility, security, and resilience, while the semantic GUI makes it easy for users to understand and edit code. This solution offers a more efficient and reliable alternative to existing methods for data transference on the internet, offering an over 90% speed improvement, and cost efficiency increase vs. traditional Web3 development methods. The lightweight design and user-friendly nature of the ARC Virtual Machine makes it easy for anyone to build and deploy their data on any EVM-compatible distributed ledger, leading to faster iteration and deployment.

GLOSSARY OF TERMS

GUI: *a visual way of interacting with a computer using items such as windows, icons, and menus, used by most modern operating systems.*

dApps: *A decentralized application is an application that can operate autonomously, typically through the use of smart contracts, that run on a decentralized computing, blockchain, or other distributed ledger system. Like traditional applications, DApps provide some function or utility to their users.*

OOP: *Object-oriented programming is a programming paradigm based on the concept of "objects", which can contain data and code. The data is in the form of fields, and the code is in the form of procedures. A common feature of objects is that procedures are attached to them and can access and modify the object's data fields.*

UML: *The Unified Modeling Language is a general-purpose, developmental modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.*

EVM: *The Ethereum Virtual Machine or EVM is a piece of software that executes smart contracts and computes the state of the Ethereum network after each new block is added to the chain. The EVM sits on top of Ethereum's hardware and node network layer.*

IDE: *An integrated development environment is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of at least a source code editor, build automation tools, and a debugger.*

Lightweight: *In computing, lightweight software also called lightweight program and lightweight application, is a computer program that is designed to have a small memory footprint and low CPU usage, overall a low usage of system resources.*

Libraries: *In computer science, a library is a collection of non-volatile resources used by computer programs, often for software development. These may include configuration data, documentation, help data, message templates, pre-written code and subroutines, classes, values, or type specifications.*

Etherscan - *Etherscan is a block explorer for the Ethereum blockchain. It allows users to easily search and browse transactions and blocks. It also provides information about each transaction and block, such as the hash and timestamp. You can think of Etherscan as the Google of Ethereum.*

Heterogenous Workflow Engine: *An extremely lightweight comprehensive development engine that Is able to work with a variety of different networks and protocols in a variety of ways. This is achieved through integrating multiple programming languages such as Solidity, Yul, and Bytecode.*

Semantic Virtual Machine: *A virtual machine designed to work with multiple different syntaxes of code such as Yul, Bytecode, and Solidity.*

Decentralized Semantic GUI - *A block explorer which displays Smart Contract information as interactive visual diagrams.*