ANASTASIA LABS

User and Developer Guides

Version 1.4 – October 2024

Contents

User and Developer Guides

1. User Guide

1.1. Introduction:

This guide provides step-by-step instructions for users to utilize the Cardano bridging contract with Mithril signatures.

1.2. Getting Started:

1.2.1. Prerequisites:

- A Cardano wallet (e.g., Nami, Eternl, Daedalus, Yoroi) with sufficient ADA.
- Access to the DApp interface (if provided).

1.2.2. Using the Bridge:

1.2.2.1. Connecting Your Wallet:

- Open the DApp and connect your Cardano wallet.

1.2.2.2. Initiating a Transfer:

- Select the destination blockchain.
- Enter the destination address, asset type and amount to transfer.

1.2.2.3. Reviewing Transaction Details:

- Verify the transaction summary displayed.
- Confirm that all details are correct.

1.2.2.4. Confirming the Transaction:

- Approve the transaction in your wallet.
- Wait for confirmation that the transaction has been submitted.

1.2.2.5. Monitoring Progress:

- Use the transaction ID to track the status.
- Receive notifications upon completion.

1.2.3. Troubleshooting:

1.2.3.1. Common Issues:

- Transaction delays due to network congestion.

- Insufficient funds or fees.

1.2.3.2. Solutions:
- Ensure adequate balance for fees.
- Retry the transaction after some time.

2. Developer Guide

2.1. Introduction:
This guide is intended for developers who wish to understand, interact with, or contribute to the bridging contract project.

2.2. Project Structure:
- Contracts Folder:
  - Contains Plutus smart contract code.
- Mithril Module:
  - Handles integration with Mithril signatures.
- UI Components:
  - Source code for the optional DApp interface.

2.3. Setting Up Development Environment:

2.3.1. Prerequisites:
- Haskell and Plutus toolchain installed.
- Access to a Cardano node or testnet environment.

2.3.2. Cloning the Repository:
- Use Git to clone the project repository.

2.3.3. Building the Project:
- Follow the build instructions provided in the README file.

2.4. Key Components and Functions:

2.4.1. Bridging Contract Functions:
- lockAssets: Function to lock assets on Cardano.
- mintTokens: Minting representation tokens on the destination chain.
- burnTokens: Burning tokens to release assets back on Cardano.

2.4.2. Mithril Integration:
- requestCertificate: Function to obtain a Mithril certificate.
- verifyCertificate: Validates the received certificate.

2.5. Testing:

2.5.1. Unit Tests:
- Located in the tests directory.
- Run using the provided test scripts.

2.5.2. Integration Tests:

- Simulate end-to-end transactions.
- Ensure proper interaction between contract and Mithril components.

2.6. Contributing Guidelines:

- Follow coding standards as per the project's style guide.

3. Submit pull requests with detailed descriptions.

- Ensure all tests pass before submitting code.