ANASTASIA LABS

HOW TO USE

Aiken Multisig Offchain

Project Number	1100025
Project Manager	Jonathan Rodriguez
Project Name:	Anastasia Labs X Maestro - Plug ‘n Play 2.0
URL:	Catalyst Proposal

Contents

How to Use the Aiken Multisig Offchain via Maestro

1. Introduction

The Aiken Multisig Offchain is designed to manage recurring payments on the Cardano blockchain. It automates subscriber-to-merchant interactions such as service creation, account registration, subscription initiation, extension, cancellation, and fund withdrawals. This guide assists developers in integrating these functions into their applications via Maestro's API endpoints.

Key features include:

- Collective approval for funds transfers
- Dynamic update of signer configurations and thresholds
- Secure termination and fund distribution upon multi-sig shutdown
- Seamless integration via RESTful API endpoints

2. Prerequisites

Before you begin, ensure that you have:

- **Access Credentials:**
  - Maestro API Key (Get from **Maestro Getting Started**)
  - Cardano wallet address with sufficient funds

- **Environment Setup:**
  - Basic familiarity with REST APIs and JSON data formats.
  - Ability to execute curl commands in a terminal/command-line environment.

3. Smart Contract Overview

The Upgradable Multi-Signature smart contract is developed using Aiken and includes:

- **Multisig NFT**: A unique non-fungible token representing the state and authority of the multi-sig wallet.
- **Datum**: Stores the list of authorized signers, the approval threshold, spending limits, and other parameters.
- **Redeemers**:
  - InitMultiSig: To create a new multi-sig setup by minting one Multisig NFT.
  - Update: To modify the signer list, threshold, and spending limits.
  - EndMultiSig: To terminate the multi-sig arrangement by burning the Multisig NFT and distributing the funds.

The contract validates that operations adhere to the specified threshold of signers and ensures that funds remain secure throughout the process.

4. MultiSig Operations

The following sections describe the three primary operations exposed via Maestro's API endpoints along with example curl commands.

4.1. Initiate MultiSig

This operation sets up a new multi-sig wallet by minting a Multisig NFT and locking funds into the multi-sig validator. The initiator supplies the initial configuration including the list of signers, required threshold, spending limits, and total funds.

Endpoint:

```
POST https://mainnet.gomaestro-api.org/v1/contracts/multisig/initiate
```

Required Parameters:

- **`initiator_address`**: Address of the entity creating the multi-sig.
- **`signers`**: An array of Cardano addresses that will be authorized to sign transactions.
- **`threshold`**: Minimum number of signatures required to approve a transaction.
- **`fund_policy_id`**: Policy ID that governs the fund asset.
- **`fund_asset_name`**: The asset name for the funds.
- **`spending_limit`**: Maximum amount that can be withdrawn in a single transaction.
- **`total_funds_qty`**: Total funds quantity to be managed by the multi-sig.

cURL Example:

```
# multisig: initiate
curl --location 'https://mainnet.gomaestro-api.org/v1/contracts/multisig/initiate' \
--header 'Content-Type: application/json' \
--header 'api-key ${API_KEY}' \
--data '{
    "initiator_address":
"addr1q9s6m9d8yedfcf53yhq5j5zsg0s58wpzamwexrxpfelgz2wgk0s9l9fqc93tyc8zu4z7hp9dlska2kew9trdg8nscjcq3sk5

    "signers": [

"addr1q9s6m9d8yedfcf53yhq5j5zsg0s58wpzamwexrxpfelgz2wgk0s9l9fqc93tyc8zu4z7hp9dlska2kew9trdg8nscjcq3sk5

"addr1qxccwptmx6r523vxcrplvfhtrpdut8s0hht0fyt9f8vy8385chtg2dupkyqu7pgqawju7awrwfg94skstmaves6hwaks6qlg
    ],
    "threshold": 2,
    "fund_policy_id": "97bbb7db0baef89caefce61b8107ac74c7a7340166b39d906f174bec",
    "fund_asset_name": "54616c6f73",
    "spending_limit": 5000,
    "total_funds_qty": 10000
}'
```

4.2. Update MultiSig

This operation allows authorized signers to update the multi-sig configuration. Changes can include modifying the signer list, adjusting the threshold, and setting a new spending limit. The update operation must be approved by the required number of signers as specified in the current configuration.

Endpoint:

```
POST https://mainnet.gomaestro-api.org/v1/contracts/multisig/update
```

Required Parameters:

- new_signers: Array of new signer addresses.
- new_threshold: The updated minimum number of signatures required.
- fund_policy_id: The policy ID for the fund asset (same as before).
- fund_asset_name: The asset name for the funds.
- new_spending_limit: Updated spending limit for transactions.

cURL Example:

```
# multisig: update
curl --location 'https://mainnet.gomaestro-api.org/v1/contracts/multisig/update' \
--header 'Content-Type: application/json' \
--header 'api-key ${API_KEY}' \
--data '{
    "new_signers": [

"addr1q9s6m9d8yedfcf53yhq5j5zsg0s58wpzamwexrxpfelgz2wgk0s9l9fqc93tyc8zu4z7hp9dlska2kew9trdg8nscjcq3sk5

"addr1qxccwptmx6r523vxcrplvfhtrpdut8s0hht0fyt9f8vy8385chtg2dupkyqu7pgqawju7awrwfg94skstmaves6hwaks6qlg

"addr1q8w9s6m9d8yedfcf53yhq5j5zsg0s58wpzamwexrxpfelgz2wgk0s9l9fqc93tyc8zu4z7hp9dlska2kew9trdg8nscjcq3s
    ],
    "new_threshold": 3,
    "fund_policy_id": "97bbb7db0baef89caefce61b8107ac74c7a7340166b39d906f174bec",
    "fund_asset_name": "54616c6f73",
    "new_spending_limit": 10000
}'
```

4.3. End MultiSig

This operation terminates the multi-sig arrangement. It requires that the required number of signers approve the termination. During this process, the Multisig NFT is burned and the remaining funds are distributed to a designated recipient.

```
POST https://mainnet.gomaestro-api.org/v1/contracts/multisig/end
```

Required Parameters:

- **`signers`**: Array of signer addresses involved in the termination (must meet the current threshold).
- **`threshold`**: Current signature threshold.
- **`fund_policy_id`**: The policy ID for the fund asset.
- **`fund_asset_name`**: The asset name for the funds.
- **`spending_limit`**: The spending limit as per the current configuration.
- **`recipient_address`**: Address where the remaining funds should be sent.

cURL Example:

```
# multisig: end
curl --location 'https://mainnet.gomaestro-api.org/v1/contracts/multisig/end' \
--header 'Content-Type: application/json' \
--header 'api-key ${API_KEY}' \
--data '{
    "signers": [

"addr1q9s6m9d8yedfcf53yhq5j5zsg0s58wpzamwexrxpfelgz2wgk0s9l9fqc93tyc8zu4z7hp9dlska2kew9trdg8nscjcq3sk5

"addr1qxccwptmx6r523vxcrplvfhtrpdut8s0hht0fyt9f8vy8385chtg2dupkyqu7pgqawju7awrwfg94skstmaves6hwaks6qlg
    ],
    "threshold": 2,
    "fund_policy_id": "97bbb7db0baef89caefce61b8107ac74c7a7340166b39d906f174bec",
    "fund_asset_name": "54616c6f73",
    "spending_limit": 5000,
    "recipient_address":
"addr1q8w9s6m9d8yedfcf53yhq5j5zsg0s58wpzamwexrxpfelgz2wgk0s9l9fqc93tyc8zu4z7hp9dlska2kew9trdg8nscjcq3s
}'
```

5. Conclusion

This guide provides a detailed walkthrough for using the Upgradable Multi-Signature smart contract via Maestro's API endpoints. By following these instructions, developers can:

- **Initiate** a new multi-sig wallet with a defined set of signers and fund limits.
- **Update** the multi-sig configuration to reflect changes in the signer list and thresholds.
- **Terminate** the multi-sig arrangement securely, ensuring proper fund distribution.

Before going live, replace all placeholder values (e.g., `${API_KEY}`, wallet addresses) with your actual data and test each command in a development (preprod/ preview) environment. For additional details, please refer to the full Design Specification Documentation and associated documentation provided by the project.