

CIP-113 Policy Manager - Quick Guide

Introduction

CIP-113 is a Cardano Improvement Proposal that defines smart tokens with programmable transfer rules. This application allows you to create, mint, and manage CIP-113 compliant tokens with built-in access control mechanisms.

Prerequisites

- **Wallet Connection:** Connect a Cardano wallet (Nami, Eternl, etc.)
 - **Admin Rights:** You must be an admin to mint tokens for a policy
 - **Network:** Currently running on Cardano Preview Testnet
-

1. Create Policy

Create a new CIP-113 policy that defines the rules and governance for your token.

Form Fields

- **Token Name*** (required): Name of your token (max 32 characters)
- **Admin Addresses:** Comma-separated list of additional admin addresses. Your wallet address is automatically included as admin
- **Enable Blacklist:** Restrict specific addresses from receiving/sending tokens
- **Enable Whitelist:** Allow only specific addresses to receive/send tokens

CREATE POLICY

POLICYMINT

Create CIP113 Policy

Token Name*

e.g. MyToken

Admin Addresses

addr1..., addr2...

☐ Enable Blacklist
Restrict specific addresses from interacting with your token

☐ Enable Whitelist
Allow only specific addresses to interact with your token

Deploy Policy

CREATE POLICY

POLICY MINT

Create CIP113 Policy

Token Name*

GeniusTestToken

Admin Addresses

addr1q8je0mh8wxjd8jd7hm74wgdajvjf0ztsyupfjshyu9u27u3mgp06jleumxth:

☒ Enable Blacklist
Restrict specific addresses from interacting with your token

☒ Enable Whitelist
Allow only specific addresses to interact with your token

Deploy Policy

Smart Contract Details

The policy creation deploys two Plutus V3 smart contracts:

1. **Rule Script** (Withdraw Validator): Enforces blacklist/whitelist rules and validates admin signatures
 - Parameterized with: blacklist linked list, whitelist linked list, admin public key hashes
2. **Smart Token Script** (Minting Policy): Controls token minting and transfer logic
 - Parameterized with: token name (hex), rule script policy hash

Transaction Flow

1. User submits form with token configuration
2. System generates both validator scripts with parameters
3. Two stake credentials are registered on-chain (one for each script)
4. Transaction is signed by user's wallet and submitted
5. Policy ID is generated from the minting script hash

Example Transaction:

Transaction URL:

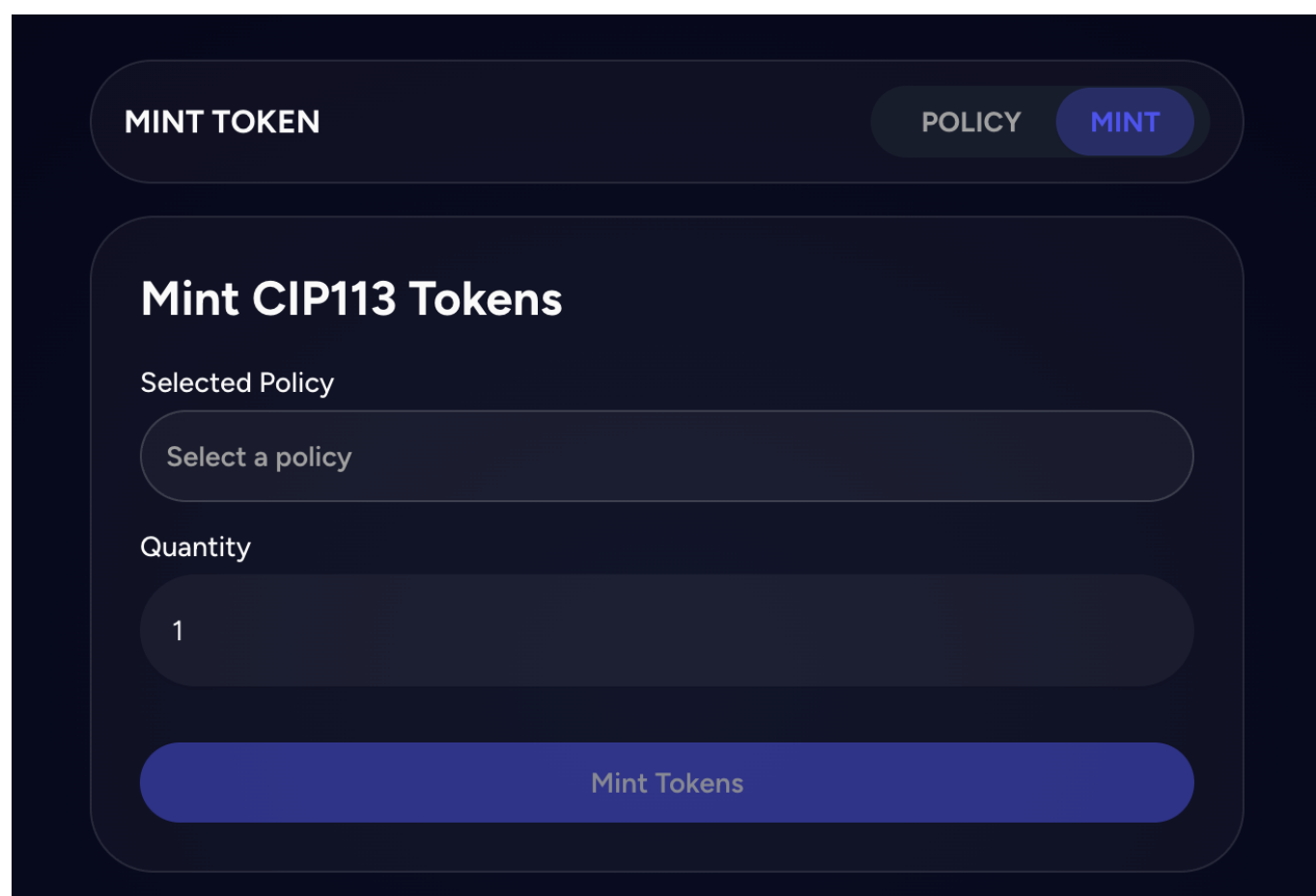
<https://preview.cexplorer.io/tx/dcacabe6e6b2b5dff1138a5b2ef21c79635ba5c0132a024fc9342efd22b4a3fb?tab=overview>

2. Mint Tokens

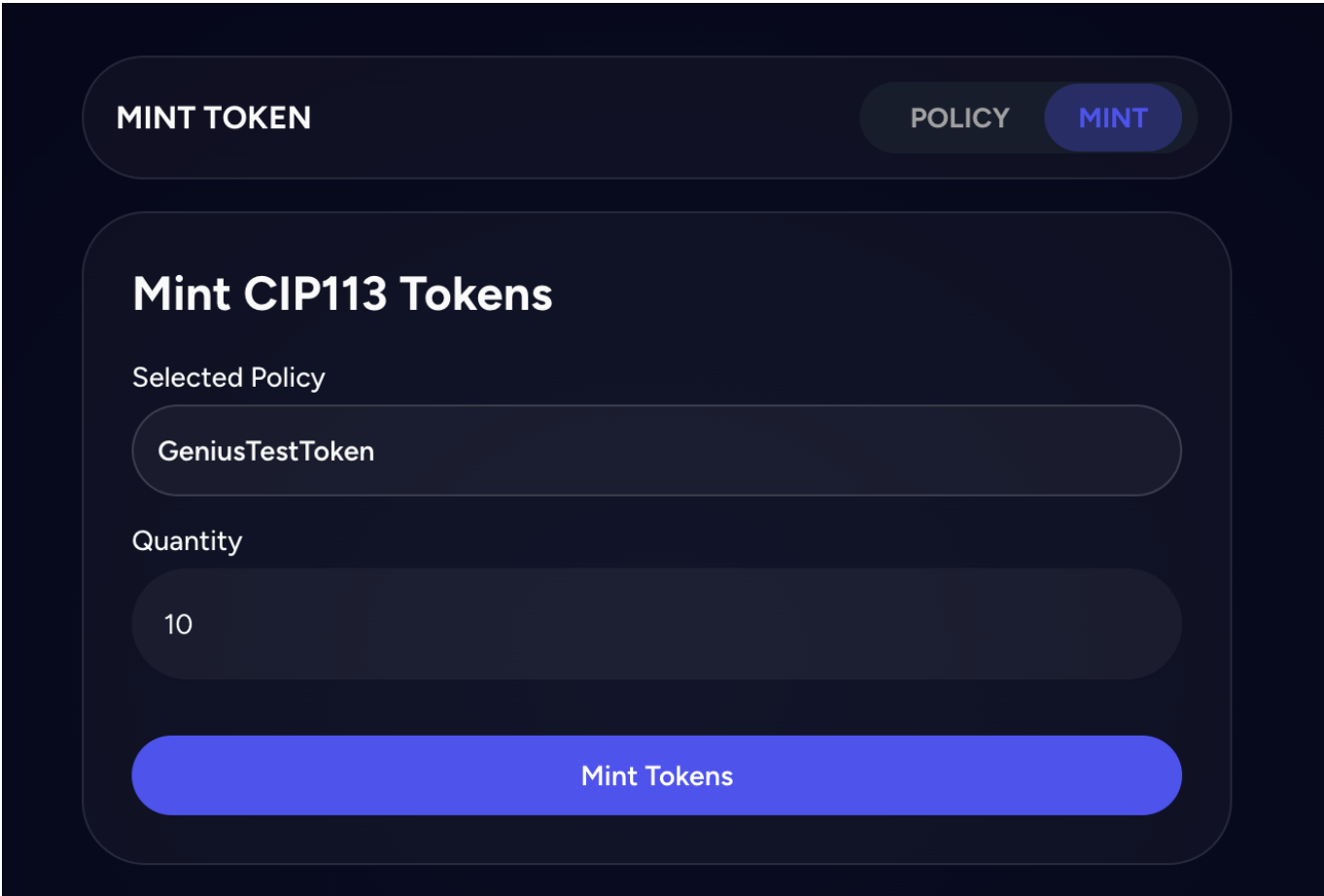
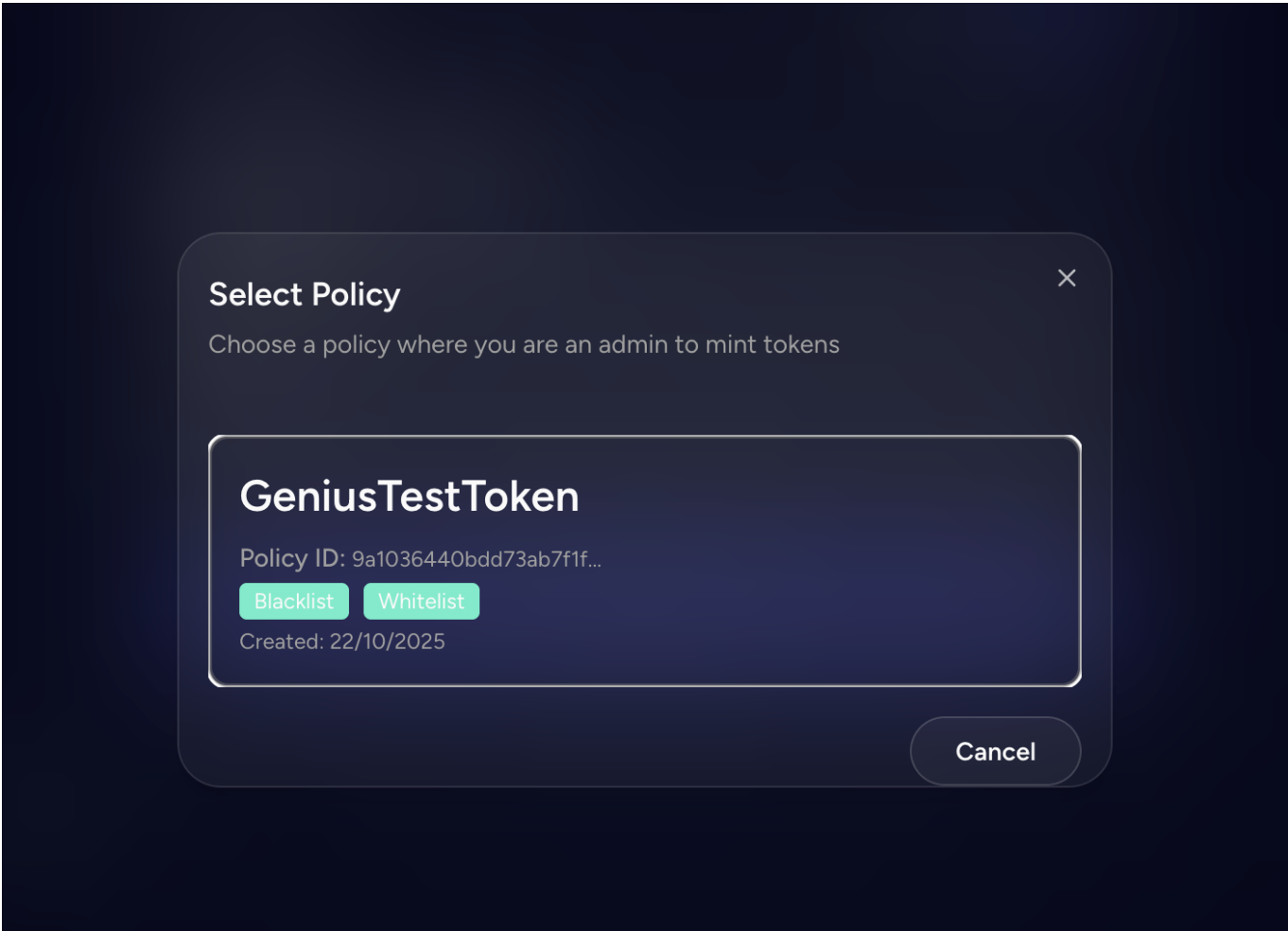
Mint tokens under an existing policy. Only admin addresses can mint tokens.

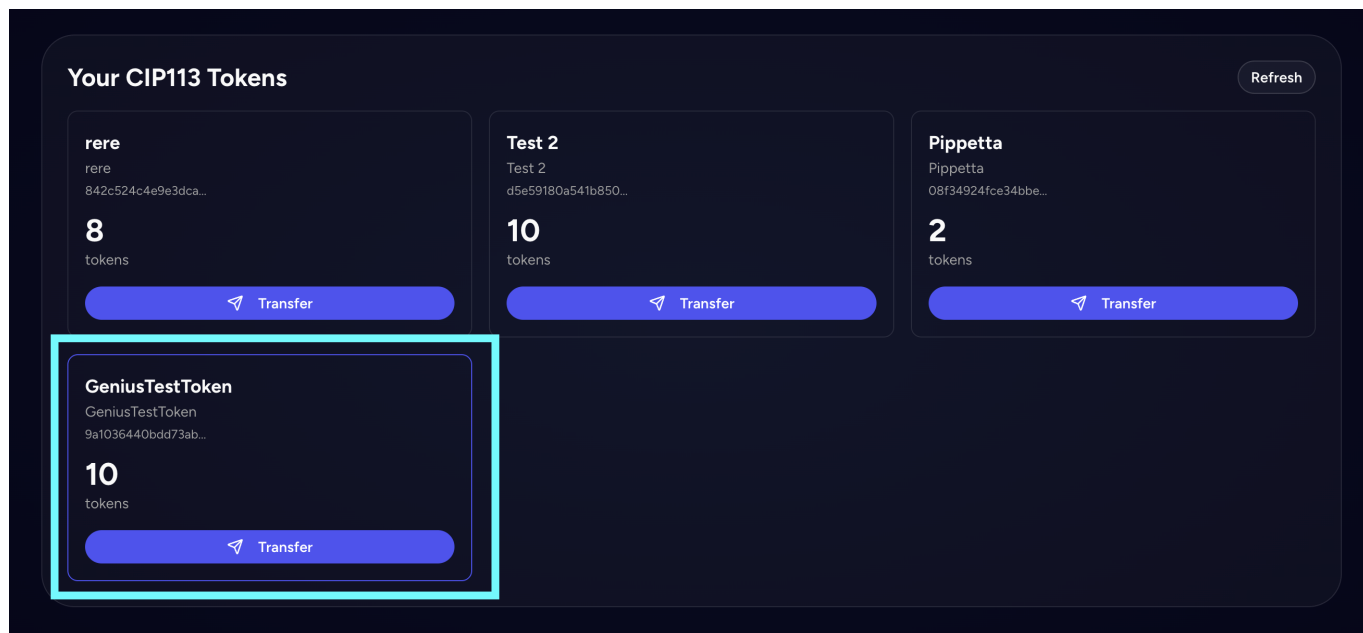
Form Fields

- **Selected Policy*** (required): Choose from policies where you are an admin
- **Quantity*** (required): Amount of tokens to mint (must be > 0)



The screenshot shows a dark-themed user interface for minting tokens. At the top, there are two tabs: 'MINT TOKEN' and 'MINT'. Below the tabs, the title 'Mint CIP113 Tokens' is displayed. Underneath the title, there are two input fields: 'Selected Policy' with a placeholder 'Select a policy' and 'Quantity' with a value of '1'. At the bottom of the form is a large blue button labeled 'Mint Tokens'.





Smart Contract Details

The minting transaction interacts with both validators:

- **Minting Script:** Validates the minting redeemer and checks rule script validation
- **Rule Script:** Executed via withdrawal (0 ADA) to verify admin signature and enforce rules

Tokens are sent to a **smart receiver address** - a script address parameterized with the user's public key hash. This ensures CIP-113 compliance and enables programmable transfer logic.

Transaction Flow

1. User selects a policy and specifies quantity
2. System calculates smart receiver address for current user
3. Transaction builds:
 - Minting output with specified quantity
 - Required signer: user's public key hash
 - Withdrawal from rule script (0 ADA) to trigger validation
 - Output sent to user's smart receiver address
4. Transaction is signed and submitted
5. Tokens appear in user's CIP-113 token balance

Example Transaction:

Transaction URL:
<https://preview.cexplorer.io/tx/3dccad56b252499e4b9d0809f9d453cb13ffcd11dc6760d81fe0616e39ef7365?tab=overview>

3. Transfer Tokens

Transfer CIP-113 tokens to another Cardano address.

Form Fields

- **Recipient Address*** (required): Destination Cardano address (addr1...)
- **Amount*** (required): Quantity to transfer (must be \leq available balance)

The modal automatically displays:

- Current token balance
- Token name and policy information
- Transfer summary with sender/recipient details

Transfer Tokens

Send CIP113 tokens to another address

Token

Policy

Available Balance

GeniusTestToken

GeniusTestToken

10

Recipient Address

addr_test1qqq0cuu96g9hny47un2qcyv7qcs3u70whcdmf06mqj3pk

Enter the Cardano address of the recipient

Amount

10

Max

You send

10

GeniusTestToken

→

They receive

10

GeniusTestToken

Recipient: addr_test1qqq0cuu96g...

Smart Contract Details

CIP-113 transfers involve spending from script addresses:

- **Smart Token Script:** Validates transfer redeemer and rule compliance
- **Rule Script:** Verifies sender signature and checks blacklist/whitelist rules

Both the sender and recipient use **smart receiver addresses** parameterized with their respective public key hashes.

Transaction Flow

1. User specifies recipient address and amount
2. System selects appropriate UTxO containing tokens
3. System calculates:
 - Recipient's smart receiver address
 - Change amount (if not transferring full balance)
4. Transaction builds:
 - Spending input: UTxO from sender's smart receiver address
 - Output 1: Tokens to recipient's smart receiver address
 - Output 2 (if change): Remaining tokens back to sender's smart receiver address
 - Withdrawals from both smart token and rule scripts (0 ADA each)
 - Required signer: sender's public key hash
5. Transaction is signed and submitted
6. Tokens are transferred to recipient's CIP-113 address

Example Transaction:

```
Transaction URL:  
https://preview.cexplorer.io/tx/418731ff5e281ce158991cd47cf607bbf35ce15d92962a4750ac43d91f6b13b2?tab=overview
```

4. Burn Tokens

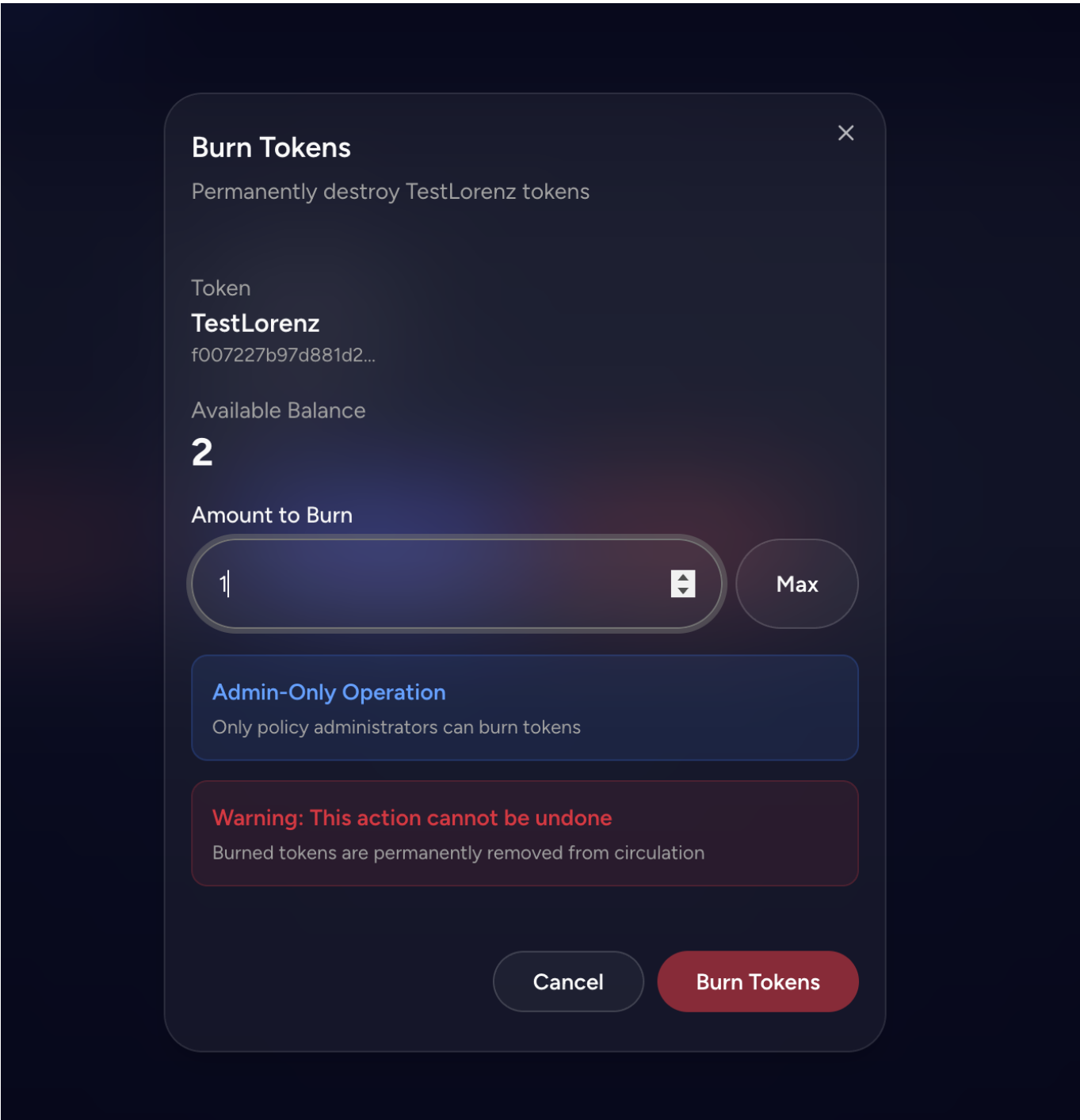
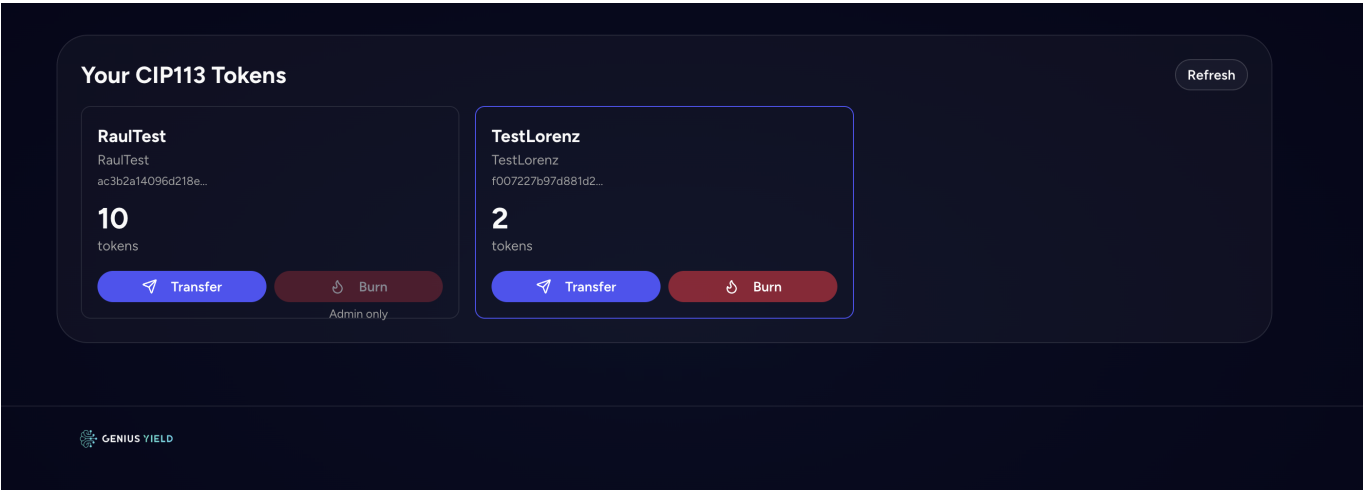
Permanently destroy CIP-113 tokens. **Only policy administrators can burn tokens.**

Form Fields

- **Amount*** (required): Quantity to burn (must be > 0 and \leq available balance)

The modal automatically displays:

- Current token balance
- Token name and policy information
- Warning message about irreversible action
- Admin-only operation notice



Important: The burn operation is restricted to policy administrators only.

- The burn button is **disabled** for non-admin users
- Only addresses listed in the policy's **adminAddresses** can execute burns
- This ensures controlled token supply management

Smart Contract Details

Token burning is implemented as negative minting:

- **Smart Token Script:** Validates burn redeemer (`ConStr0(["burn"])`) and rule compliance
- **Rule Script:** Verifies admin signature and enforces governance rules
- **Minting Amount:** Negative value (e.g., **-100** to burn 100 tokens)

The transaction spends UTxOs from the admin's smart receiver address and burns the specified amount.

Transaction Flow

1. Admin user specifies amount to burn
2. System validates user is in policy's admin list
3. System selects appropriate UTxO(s) containing tokens
4. Transaction builds:
 - Spending input: UTxO(s) from admin's smart receiver address
 - Negative mint: Burns specified token quantity
 - Change output (if applicable): Remaining tokens back to admin's smart receiver address
 - Withdrawals from both smart token and rule scripts (0 ADA each)
 - Required signer: admin's public key hash
5. Transaction is signed and submitted
6. Tokens are permanently removed from circulation

Example Transaction:

```
Transaction URL:  
https://preview.cexplorer.io/tx/95921989c3255ed28698f6dcc6ef27f5dee448bce7  
14f9ee98c373ca181bce29
```

Important Notes

- **Irreversible:** Burned tokens cannot be recovered
- **Admin Only:** Only policy admins can execute burn transactions
- **Supply Reduction:** Burning permanently reduces total token supply
- **Change Handling:** If burning partial balance, change is returned to admin's smart receiver address

Technical Notes

- **Script Addresses:** All CIP-113 tokens are held in script addresses (smart receiver addresses), not regular wallet addresses

- **Collateral:** Transactions involving smart contracts require collateral UTxOs
- **Plutus Version:** All scripts are Plutus V3
- **Redeemers:**
 - Minting uses `ConStr0(["mesh"])`
 - Burning uses `ConStr0(["burn"])`
 - Transfers use `ConStr0([])` for token script and `ConStr1([[signerHash], [0], [0]])` for rule validation
 - Spending from smart addresses uses `ConStr0([])`