TxPipe Docs

Asset Expressions

This guide covers how to work with blockchain assets and values in Tx3.

Overview

Tx3 provides built-in support for working with blockchain assets:

- Native currency (ADA)
- Custom tokens
- NFTs
- Multi-asset bundles

Asset Definitions

Native Asset (ADA)

```
// ADA is built-in
Ada(1000000)  // 1 ADA
Ada(500000)   // 0.5 ADA
```

Custom Assets

```
// Define custom asset
asset MyToken = "policy_id" "asset_name";

// Use custom asset
MyToken(100)
```

AnyAsset Catch-All Constructor

```
// Define custom asset and amount
AnyAsset("policy_id", "asset_name", <amount_int>)

// Define NFT
AnyAsset("policy_id", "asset_name", 1)
```

Asset Bundles

```
// Combine multiple assets
Ada(1000000) + MyToken(50)

// Complex bundles
Ada(1000000) + MyToken(50) + AnyAsset("policy_id", "asset_name", 1)
```

Asset Expressions

Basic Operations

```
// Addition
asset1 + asset2

// Subtraction
asset1 - asset2

// Property access
asset.amount
```

Examples

```
// Simple transfer
tx transfer(amount: Int) {
    input source {
        from: Sender,
        min_amount: Ada(amount),
    }

    output {
        to: Receiver,
        amount: Ada(amount),
    }
}

// Multi-asset transfer
tx transfer_multi(
    ada_amount: Int,
    token_amount: Int
) {
    input source {
        from: Sender,
        min_amount: Ada(ada_amount) + MyToken(token_amount),
    }

    output {
        to: Receiver,
        amount: Ada(ada_amount) + MyToken(token_amount),
    }
}
```

Asset Validation

Minimum Amounts

```
input source {
    from: Sender,
    min_amount: Ada(1000000),  // At least 1 ADA
}
```

Exact Amounts

```
output {
    to: Receiver,
    amount: Ada(1000000),  // Exactly 1 ADA
}
```

Asset Combinations

```
input source {
    from: Sender,
    min_amount: Ada(1000000) + MyToken(50),  // Multiple assets
}
```

Common Patterns

Token Swap

```
tx swap(
    input_amount: Int,
    output_amount: Int
) {
    input source {
        from: User,
        min_amount: TokenA(input_amount),
    }

    output {
        to: User,
        amount: TokenB(output_amount),
    }
}
```

NFT Transfer

```
tx transfer_nft(
    policy_id: Bytes,
    asset_name: Bytes
) {
    input source {
        from: Owner,
        min_amount: AnyAsset(policy_id, asset_name, 1),
    }

    output {
        to: Recipient,
        amount: AnyAsset(policy_id, asset_name, 1),
    }
}

### FT Transfer
```tx3
tx transfer_nft(
    policy_id: Bytes,
    asset_name: Bytes,
    amount: Int
) {
    input source {
        from: Owner,
        min_amount: AnyAsset(policy_id, asset_name, amount),
    }

    output {
        to: Recipient,
        amount: AnyAsset(policy_id, asset_name, amount),
    }
}
```
```

Liquidity Pool

```
tx add_liquidity(
    ada_amount: Int,
    token_amount: Int
) {
    input ada {
        from: User,
        min_amount: Ada(ada_amount),
    }

    input token {
        from: User,
        min_amount: MyToken(token_amount),
    }

    output {
        to: Pool,
        amount: Ada(ada_amount) + MyToken(token_amount),
    }
}
```

Common Use Cases

Token Minting

```
tx mint_tokens(
    amount: Int
) {
    input source {
        from: MintingAuthority,
        min_amount: Ada(fees),
    }

    mint {
        amount: MyToken(amount),
        redeemer: MintData { quantity: amount },
    }
}
```

Token Burning

```
tx burn_tokens(
    amount: Int
) {
    input source {
        from: User,
        min_amount: MyToken(amount),
    }

    burn {
        amount: MyToken(amount),
        redeemer: BurnData { quantity: amount },
    }
}
```

Asset Locking

```
tx lock_assets(
    amount: Int
) {
    input source {
        from: User,
        min_amount: Ada(amount) + MyToken(amount),
    }

    output locked {
        to: LockContract,
        amount: Ada(amount) + MyToken(amount),
        datum: LockState {
            amount: amount,
            owner: User,
        }
    }
}
```