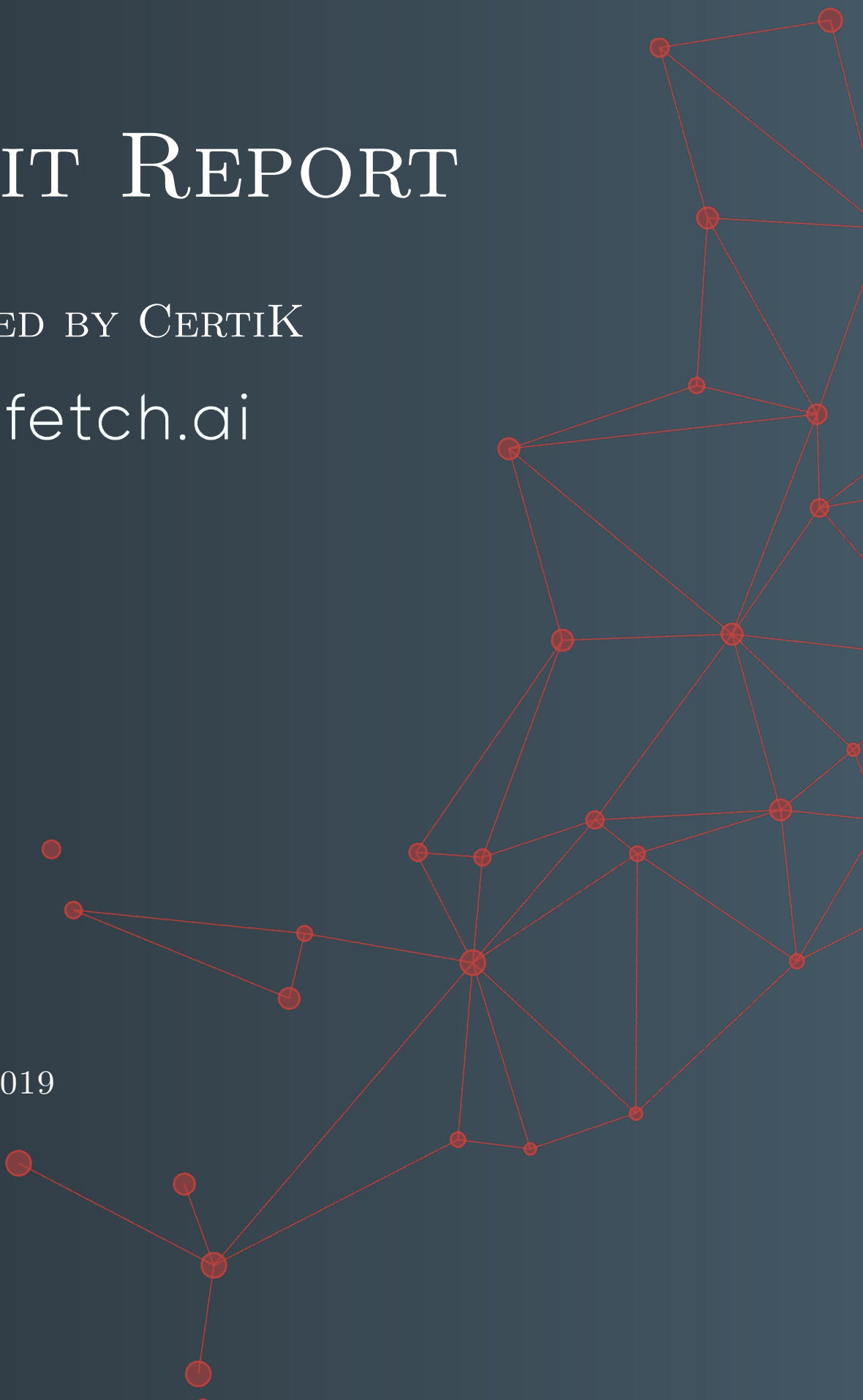
CERTIK

Audit Report

Produced by CertiK

for fetch.ai



13th Dec, 2019

CertiK Audit Report
For Fetch.AI

Request Date: 2019-11-20
Revision Date: 2019-12-13
Platform Name: Ethereum

Contents

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and Fetch.AI(the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the "Agreement"). This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, has developed a proprietary Formal Verification technology to apply rigorous and complete mathematical reasoning against code. This process ensures algorithms, protocols, and business functionalities are secured and working as intended across all platforms.

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis, and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 6.2B in assets.

For more information: https://certik.org/

Executive Summary

This report has been prepared for Fetch.AI to discover issues and vulnerabilities in the source code of their NativeTokenMigration smart contracts. A comprehensive examination has been performed, utilizing CertiK's Formal Verification Platform, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Vulnerability Classification

CertiK categorizes issues into three buckets based on overall risk levels:

Critical

Code implementation does not match specification, which could result in the loss of funds for contract owner or users.

Medium

Code implementation does not match the specification under certain conditions, which could affect the security standard by loss of access control.

Low

Code implementation does not follow best practices, or uses suboptimal design patterns, which could lead to security vulnerabilities further down the line.

Testing Summary

PASS

CERTIK *believes this smart contract passes security qualifications to be listed on digital asset exchanges.*

Dec 13, 2019

Type of Issues

CertiK's smart label engine applied 100% formal verification coverage on the source code. Our team of engineers has scanned the source code using proprietary static analysis tools and code-review methodologies. The following technical issues were found:

Title	Description	Issues	SWC ID
Integer Overflow/ Underflow	An overflow/underflow occurs when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function Incorrectness	Function implementation does not meet specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker can write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by miners to some degree.	3	SWC-116
Insecure Compiler Version	Using a fixed outdated compiler version or floating pragma can be problematic if there are publicly disclosed bugs and issues that affect the current compiler version used.	0	SWC-102 SWC-103
Insecure Randomness	Using block attributes to generate random numbers is unreliable, as they can be influenced by miners to some degree.	0	SWC-120
"tx.origin" for Authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115

Title	Description	Issues	SWC ID
Delegatecall to Untrusted Callee	Calling untrusted contracts is very dangerous, so the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default, meaning a malicious user can make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized Variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used.	0	SWC-111
Unused Variables	Unused variables reduce code quality	0	SWC-131

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Manual Review Notes

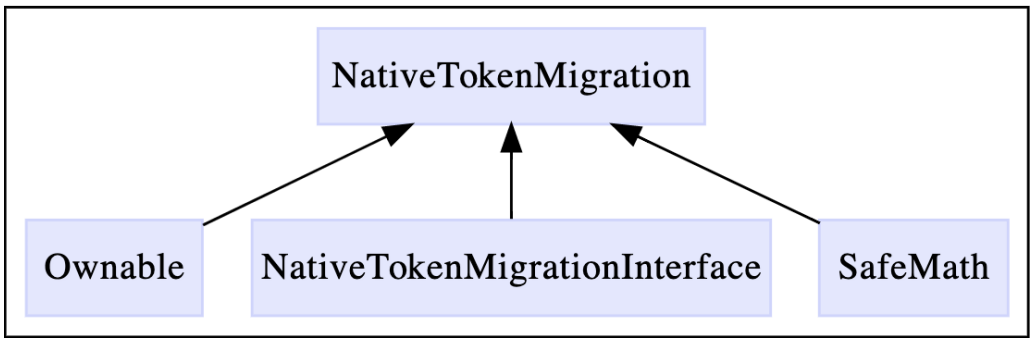
Source Code SHA-256 Checksum

- **NativeTokenMigration.sol**
  `41e737e21fdc3429fa4af11503816b322c1d8dd59310f021697322b7be2c7f5e`
- **ERC20.sol**
  `67e6e9f8bd10bc55c97a1a55c214524864920a07abe3646449fd8de434a4ba29`
- **IERC20.sol**
  `23221a896472eeee23d71500d71f40bcce31112b9198389310d2e7ff7d0be093`
- **Ownable.sol**
  `4857ce63c07e3ec7bed1b96507666bb671fa5c6df7c87750754fac8ec640c9db`
- **SafeMath.sol**
  `469b57d4f3c4e1d39e117ea6839a987e2a6e5b2fde6cce7a72e609df8b7b1443`

Summary

CertiK worked closely with Fetch.Ai to audit the design and implementation of its soon-to-be released smart contract. To ensure comprehensive protection, the source code was analyzed by the proprietary CertiK formal verification engine and manually reviewed by our smart contract experts and engineers. That end-to-end process ensures proof of stability as well as a hands-on, engineering-focused process to close potential loopholes and recommend design changes in accordance with best practices.

Fetch.Ai Architect & Workflow Overview



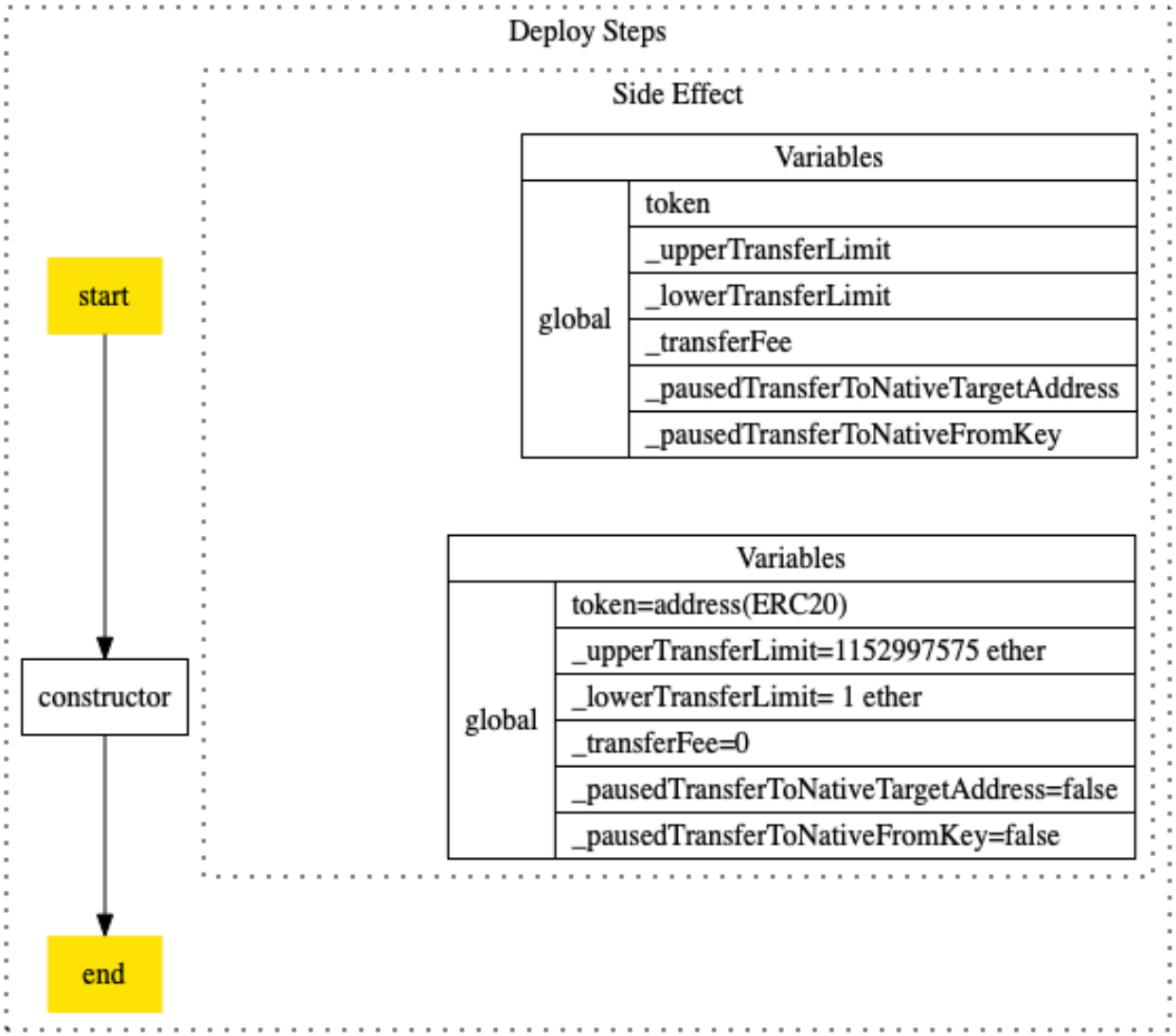
Global State Variables

Variable Name	Constant	Initialised By Func	Modified By Func
`DELETE_PERIOD`	✓		
`FET_TOTAL_SUPPLY`	✓		
`FET_MULTIPLIER`	✓		
`globalSwapID`			`_initSwap`
`swaps`			`_initSwap`,`refund`,`_reject`
`token`		`constructor`	
`delegates`			`setDelegate`
`_upperTransferLimit`		`constructor`	`setUpperTransferLimit`
`_lowerTransferLimit`		`constructor`	`setLowerTransferLimit`
`_completedAmount`			`_initSwap`,`_reject`,`withdrawToFoundation`
`_earliestDelete`			`_initSwap`
`_transferFee`			`_initSwap`, `setTransferFee`
`_pausedTransferToNativeTargetAddress`		`constructor`	`pauseTransferToNativeTargetAddress`
`_pausedTransferToNativeFromKey`		`constructor`	`pauseTransferToNativeFromKey`

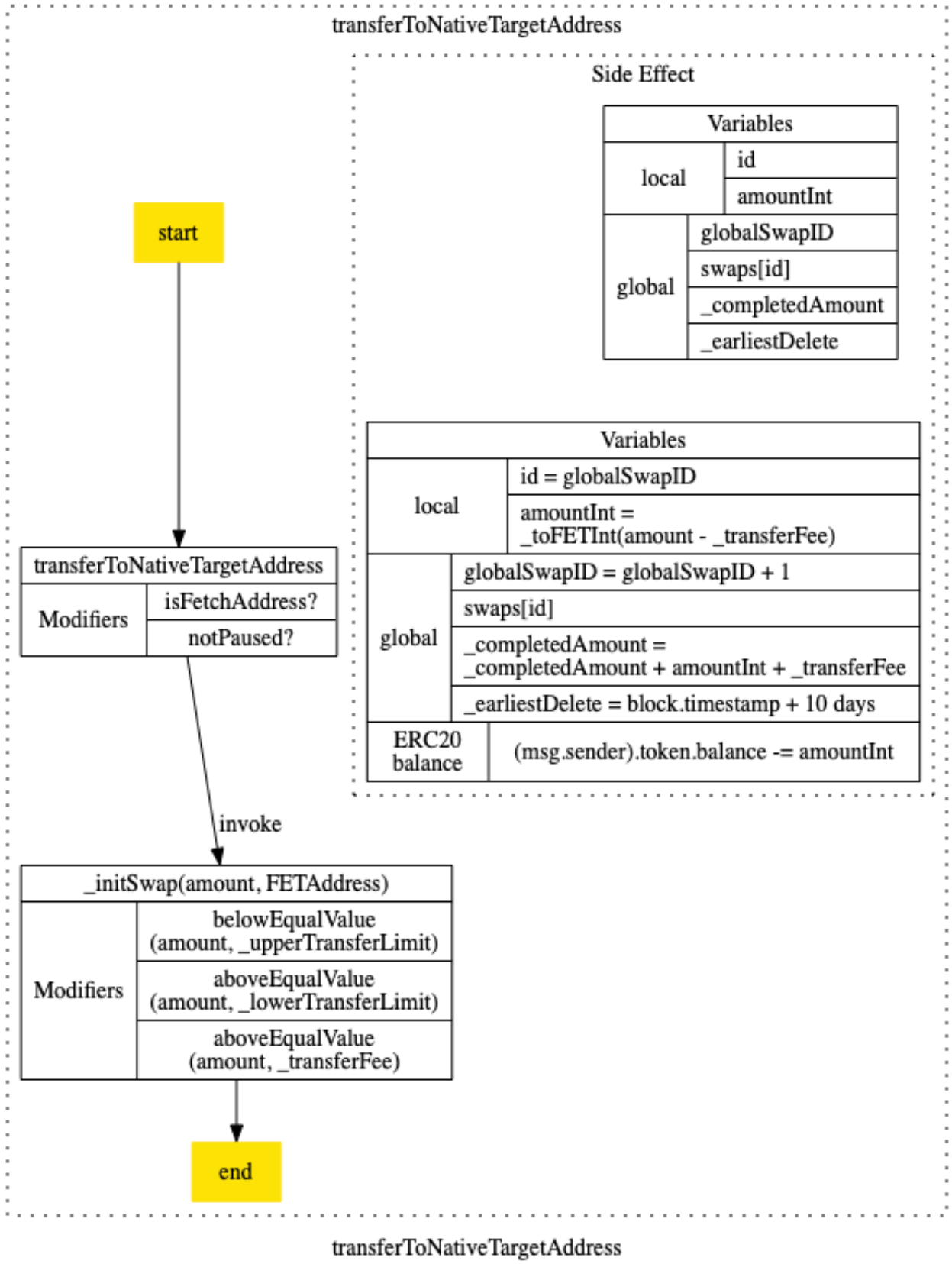
Function Access Roles

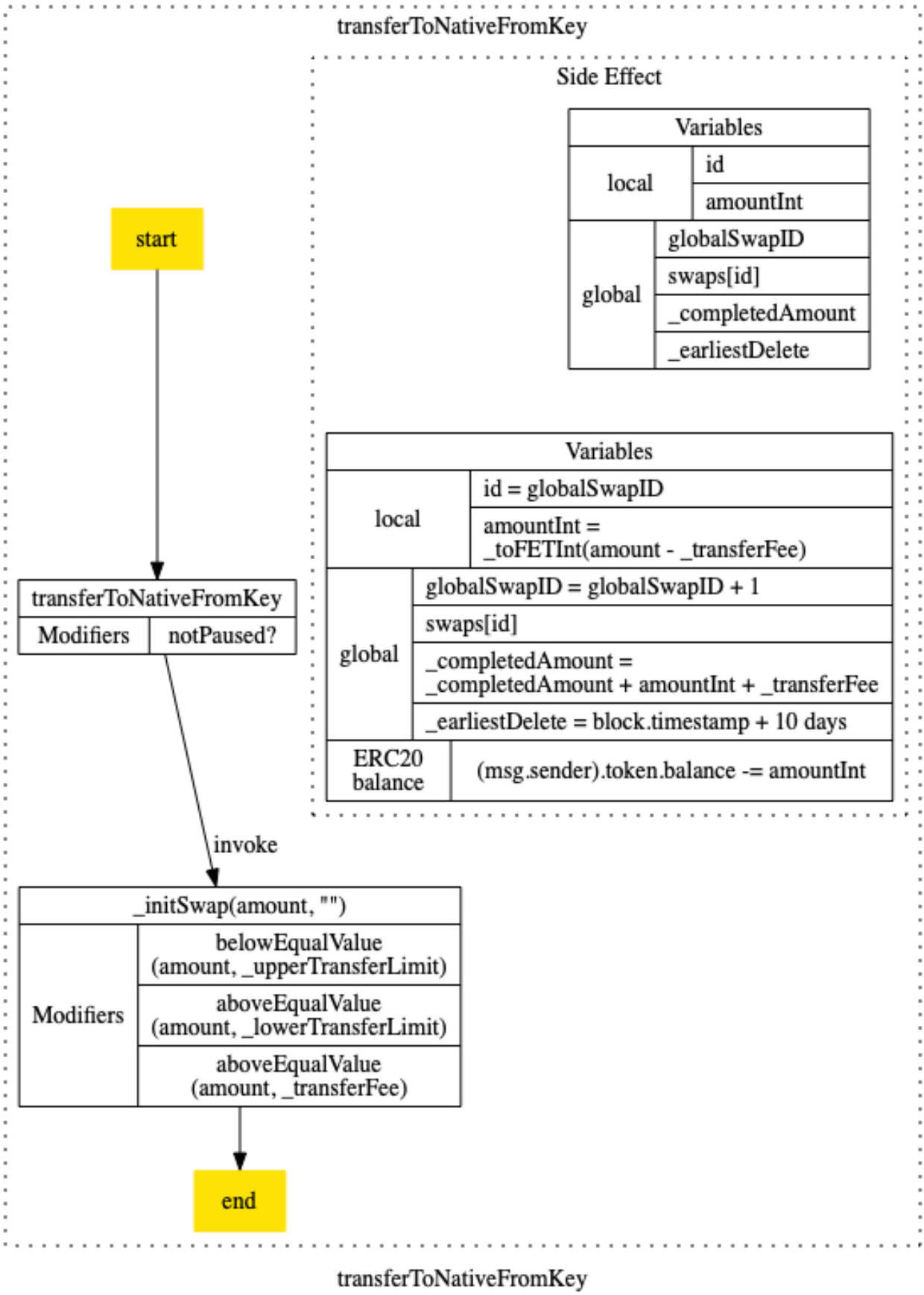
Function	isFetchAddress	isSender	onlyOwner	onlyDelegate	notPaused	isRefundable	isInitialised
`transferToNativeTargetA`	✓				✓		
`transferToNativeFromKe`					✓		
`refund`		✓				✓	
`requestRefund`		✓					✓
`pauseTransferToNativeT`				✓			
`pauseTransferToNativeF`				✓			
`setDelegate`			✓				
`setUpperTransferLimit`			✓				
`setLowerTransferLimit`			✓				
`setTransferFee`			✓				
`_reject`							✓
`reject`				✓			
`batchReject`				✓			
`withdrawToFoundation`			✓				
`deleteContract`			✓				

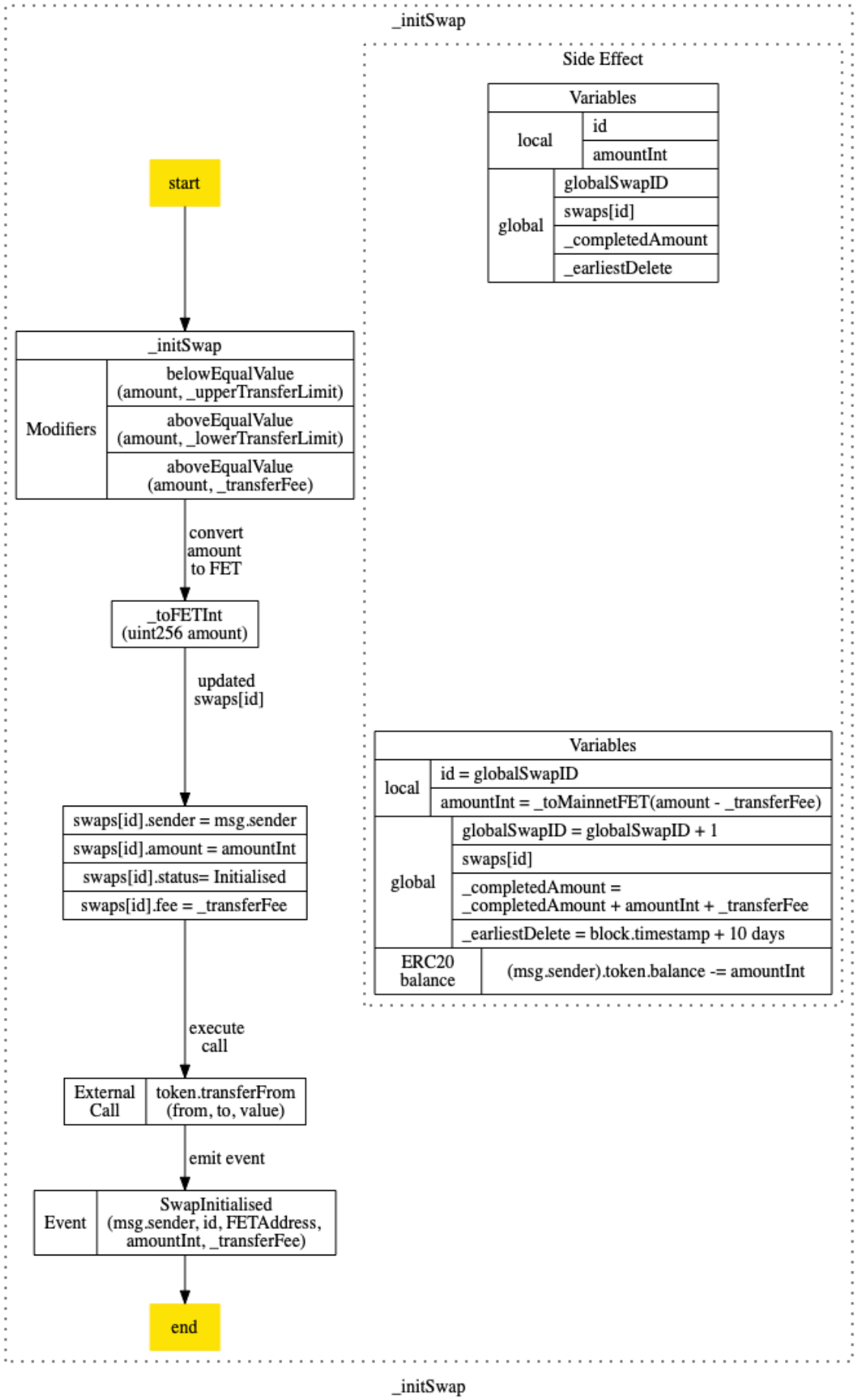
Function & Variable Analysis

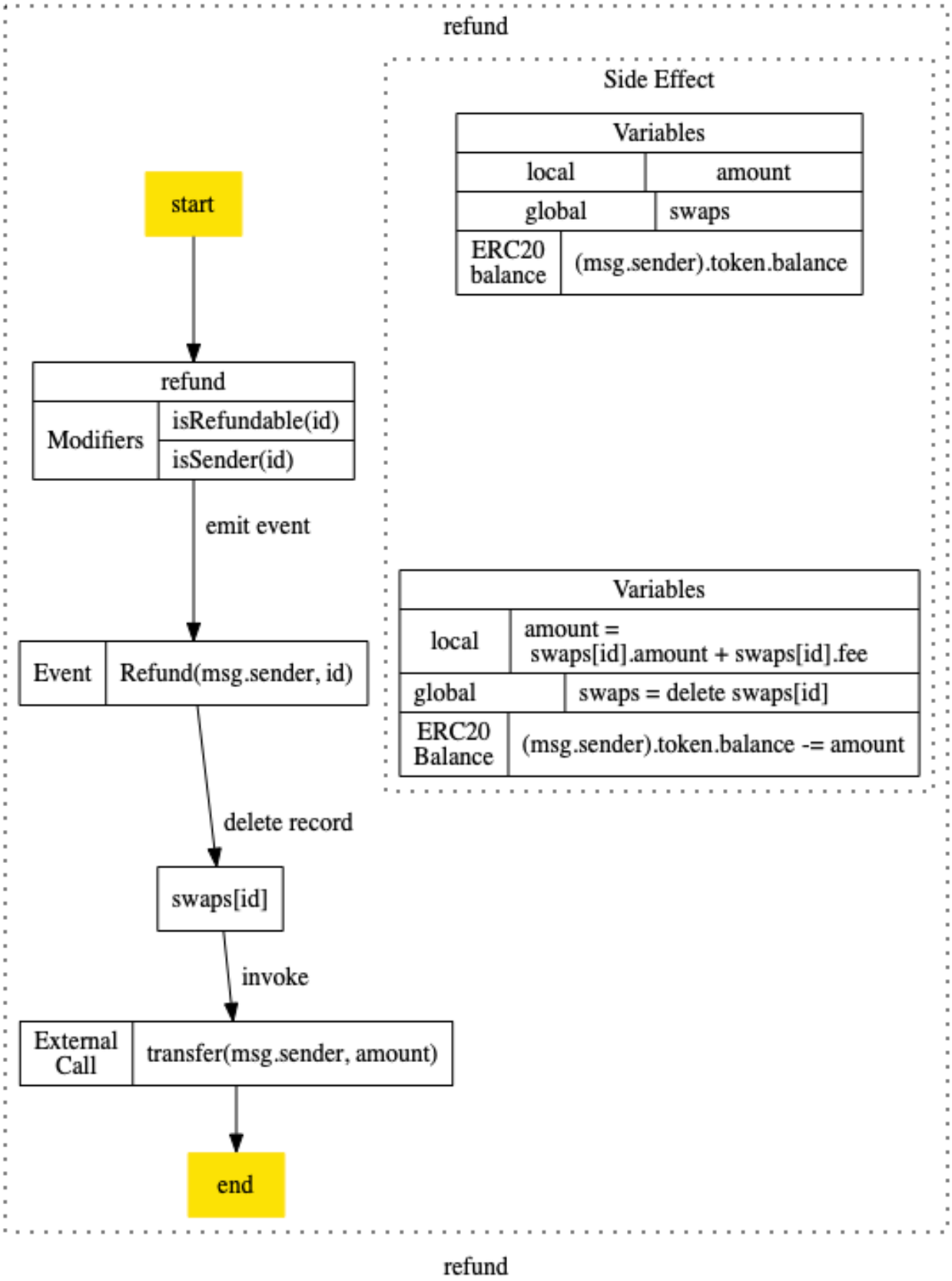


Deploy Steps Variable Analysis


transferToNativeTargetAddress
Side Effect
Variables
local
id
amountInt
global
globalSwapID
swaps[id]
_completedAmount
_earliestDelete
start
Variables
local
id = globalSwapID
amountInt =
_toFETInt(amount - _transferFee)
global
globalSwapID = globalSwapID + 1
swaps[id]
_completedAmount =
_completedAmount + amountInt + _transferFee
_earliestDelete = block.timestamp + 10 days
ERC20
balance
(msg.sender).token.balance -= amountInt
transferToNativeTargetAddress
Modifiers
isFetchAddress?
notPaused?
invoke
_initSwap(amount, FETAddress)
Modifiers
belowEqualValue
(amount, _upperTransferLimit)
aboveEqualValue
(amount, _lowerTransferLimit)
aboveEqualValue
(amount, _transferFee)
end
transferToNativeTargetAddress


transferToNativeFromKey
Side Effect
Variables
local
id
amountInt
global
globalSwapID
swaps[id]
_completedAmount
_earliestDelete
start
Variables
local
id = globalSwapID
amountInt =
_toFETInt(amount - _transferFee)
global
globalSwapID = globalSwapID + 1
swaps[id]
_completedAmount =
_completedAmount + amountInt + _transferFee
_earliestDelete = block.timestamp + 10 days
ERC20
balance
(msg.sender).token.balance -= amountInt
transferToNativeFromKey
Modifiers
notPaused?
invoke
_initSwap(amount, "")
Modifiers
belowEqualValue
(amount, _upperTransferLimit)
aboveEqualValue
(amount, _lowerTransferLimit)
aboveEqualValue
(amount, _transferFee)
end
transferToNativeFromKey


_initSwap
Side Effect
Variables
local
id
amountInt
global
globalSwapID
swaps[id]
_completedAmount
_earliestDelete
start
_initSwap
Modifiers
belowEqualValue
(amount, _upperTransferLimit)
aboveEqualValue
(amount, _lowerTransferLimit)
aboveEqualValue
(amount, _transferFee)
convert
amount
to FET
_toFETInt
(uint256 amount)
updated
swaps[id]
swaps[id].sender = msg.sender
swaps[id].amount = amountInt
swaps[id].status= Initialised
swaps[id].fee = _transferFee
execute
call
External
Call
token.transferFrom
(from, to, value)
emit event
Event
SwapInitialised
(msg.sender, id, FETAddress,
amountInt, _transferFee)
end
Variables
local
id = globalSwapID
amountInt = _toMainnetFET(amount - _transferFee)
global
globalSwapID = globalSwapID + 1
swaps[id]
_completedAmount =
_completedAmount + amountInt + _transferFee
_earliestDelete = block.timestamp + 10 days
ERC20
balance
(msg.sender).token.balance -= amountInt
_initSwap


refund
Side Effect
Variables
local
amount
global
swaps
ERC20 balance
(msg.sender).token.balance
start
refund
Modifiers
isRefundable(id)
isSender(id)
emit event
Event
Refund(msg.sender, id)
Variables
local
amount = swaps[id].amount + swaps[id].fee
global
swaps = delete swaps[id]
ERC20 Balance
(msg.sender).token.balance -= amount
delete record
swaps[id]
invoke
External Call
transfer(msg.sender, amount)
end
refund

Recommendations

Items in this section are not critical to the overall functionality of Fetch.Ai's smart contracts; however, we leave it to the client's discretion to decide whether to address them before the final deployment of source codes. Recommendations are labeled CRITICAL, MAJOR, MINOR, INFO, and DISCUSSION in decreasing significance level.

Github Review Round 1: 31c54bd2556d8383ae35fa1d0eabdec8f001f8d9

Github Review Round 2: 624bf968fdd6946d20249348c93573e163c3dc1d

Github Review Round 3: f988005a9512a7245bfbd05a1a396f35a8a3e000

Github Review Round 4: 1f8438004f9b310ed59a31e1e756497cf1c7f5ab

NativeTokenMigration.sol
INFO `_toMainnetFET()`: Recommend removing the return value `amountInt` if not used.

- ✓ Fetch.Ai Accepted, the code is updated and reflected in the latest commit.

DISCUSSION `_initSwap()`: Is the sum of `amount` always smaller than `FET_TOTAL_SUPPLY`? If not, `_completedAmount + amountInt + _transferFee` could be greater than `FET_TOTAL_SUPPLY`. Is it necessary to track the amount of FET that has already been used when trying to init a new swap?

- ✓ Fetch.Ai Confirmed, the backend does a full checksum check as specified herehttps://-docs.fetch.ai/etch-language/addresses/. The main reason this is not done within the contract itself is that it requires a base58 decoder. We decided that implementing this functionality on-chain would (i) introduce additional complexity and thereby technical risk (ii)increase the cost of initialising a swap unnecessary

INFO `_initSwap()` and `_reject()`: Recommend using `SafeMath` for any arithmetic operations.

- ✓ Fetch.Ai Accepted, the code is updated and reflected in the latest commit.

DISCUSSION `setUpperTransferLimit()`: Should we add a modifier `aboveEqualValue(newLimit, _lowerTransferLimit)` to prevent the `newLimit` lower than the current `_lowerTransferLimit`?

- ✓ Fetch.Ai Accepted, the code is updated and reflected in the latest commit.

DISCUSSION `setLowerTransferLimit()`: Recommend emitting an event log.

- ✓ Fetch.Ai Accepted, the code is updated and reflected in the latest commit.

DISCUSSION `setTransferFee()`: Recommend emitting an event log.

- ✓ Fetch.Ai Accepted, the code is updated and reflected in the latest commit.

DISCUSSION For the address check in `modifier isFetchAddress()`, what kind of checks would be performed on the backend? What if a fake address with correct length were to pass the modifier check?

- ✓ Fetch.Ai Confirmed that the amount cannot possibly be larger than `FET_TOTAL_SUPPLY` as it is coupled with a token transfer of `amountInt + _transferFee`. As there only exist `FET_TOTAL_SUPPLY` tokens, no transfer of FET tokens above that would increase it past this value can ever succeed.

DISCUSSION What are the use-cases where the function `withdrawToFoundation()` will be triggered during the native token migration process?

- ✓ Fetch.Ai `withdrawToFoundation()` will not be called automatically. Its role is to allow the migrated tokens to be transferred to different addresses from time to time, for two reasons:

  1. to reduce the overall balance contained within the contract to mitigate risks in the case of a bug in the NativeTokenMigration contract
  2. be enable the transfer of tokens in the opposite direction which is planned to be implemented in the near future.

DISCUSSION Shall `_initSwap()` only be executed once? The `_earliestDelete` is a global variable that will reset every time when this function is invoked, hence our question

- ✓ Fetch.Ai initSwap() will be executed upon every initialisation of a new swap, but should only be executable once per swap. `_earliestDelete` is reset upon each swap to ensure that the foundation cannot delete the contract before users have had time to request refunds for any open rejected swaps.

  – Arguably because the rejection of a swap depends on the foundation processing it this still leaves some potential for the swaps not being processed during that time-frame.

  – However, since the Fetch mainnet is not observable from ethereum there does not seem a simple way of preventing this. Nonetheless this appears to be a reasonable safeguard against human error and if we assume that the swaps will be processed reasonably quickly provides protection for the user from the point at which there swap is rejected.

  – In the case where we want to delete the contract the plan is to first pause the initialisation of new swaps (meaning `_earliestDelete` will also no longer increase), and to then delete the contract after everything has been processed correctly

DISCUSSION There might be “precision” issues with the `refund()` while converting to FET in `_toFetInt()`. The original amount is rounded in `_initSwap()` from 1.8 to 1.0 FET. Was this implemented by design or an error that should be addressed?

- ✓ Fetch.Ai This is intentional to simplify the migration process to mainnet, which has fewer decimal places than the ERC20 token. We have now changed this to use all if the precision that is present on the mainnet (10 decimals vs the ERC20's 18 decimals) by changing `FET_MULTIPLIER` to `DECIMAL_DIFFERENTIATOR = 10**8`

DISCUSSION In `setUpperTransferLimit()` Recommend adding check for new newLimit cannot exceed the `FET_TOTAL_SUPPLY`. Also could the newLimit be a number that lower the current one or could it be a number set by fetch.ai?

- ✓ Fetch.Ai We have added the check. The new limit might be lower than is set currently, so it is fully within the foundations discretion to set this value appropriately

DISCUSSION `setLowerTransferLimit()` Recommend adding check for new `newLimit` to make sure it is not exceeding the `FET_TOTAL_SUPPLY`. Also could the newLimit be a number that is higher than the current one?

- ✓ Fetch.Ai Added the check. Same as above.

DISCUSSION Recommend removing any unused contracts in ERC20 folder. Given the NativeTokenMigration, primarily only using the common ERC20 function for interacting the account balance.

- ERC20Burnable.sol
- ERC20Capped.sol
- ERC20Detailed.sol
- ERC20Mintable.sol
- ERC20Pausable.sol
- TokenTimelock.sol
- ✓ Fetch.Ai Removed the suggested files.

INFO Consider defining below constants using default `Time Units` for increasing the readability and maintainability of the code. Would the fact that 2020 is a leap year have any impact on your business decisions/logic.

- `DELETE_PERIOD` as `10 days`
- `FET_TOTAL_SUPPLY` as `1152997575 ether`
- `FET_MULTIPLIER` as `1 ether`
- ✓ Fetch.Ai We have changed the `DELETE_PERIOD` definition to use units of days. We're not sure if using ether as a unit for `FET_TOTAL_SUPPLY`, `FET_MULTIPLIER` would be clearer as it would require readers to know that ether and FET use the same number of digits. We have decided to changed the units to `1152997575 \cdot 10 ** 18` and `1 \cdot 10 ** 18` instead.

INFO `onlyDelegate()` Recommend renaming the function to `onlyDelegators()`.

- ✓ Fetch.Ai A delegator is the person that delegates some capability or responsibility to someone else while a delegate is the person that receives this capability. In this case, the naming of the function appears to be consistent with correct usage of these terms.

INFO `isSender()` Recommend renaming the function to `onlySender()`.

- ✓ Fetch.Ai Accepted, the code is updated and reflected in the latest commit.

INFO `notPaused()` Recommend renaming the function to `whenNotPaused()`.

- ✓ Fetch.Ai Accepted, the code is updated and reflected in the latest commit.

INFO `isRefundable()` Recommend renaming the function to `isRejected()`. Error message is not providing enough/specific context. i.e `The swap status has been rejected`.

- ✓ Fetch.Ai Renamed and changed error message to "Swap is not rejected". Also changed the error message of isInitialised() to "Swap is not initialised"

INFO Consider `setDelegate()` is an importance variable, we recommend emitting an event `ChangeDelegators()` for logging and tracking.

- ✓ Fetch.Ai Accepted, the code is updated and reflected in the latest commit.

INFO Given `setUpperTransferLimit()` is an importance variable, we recommend emitting an event `ChangeUpperTransferLimit()` for logging and tracking.

- ✓ Fetch.Ai Accepted, the code is updated and reflected in the latest commit.

INFO Given `setLowerTransferLimit()` is an importance variable, we recommend emitting an event `ChangeLowerTransferLimit()` for logging and tracking.

- ✓ Fetch.Ai Accepted, the code is updated and reflected in the latest commit.

MINOR Given `deleteContract()` is an importance function, we recommend emitting an event for logging.

- ✓ Fetch.Ai Accepted, the code is updated and reflected in the latest commit.

INFO `_reject()` Recommend emitting an event log for history tracking.

- ✓ Fetch.Ai The code is updated and reflected in the latest commit.

INFO `withdrawToFoundation()` Recommend emitting an event log for history tracking.

- ✓ Fetch.Ai Added event, the code is updated and reflected in the latest commit.

Best Practice

Smart contract development requires a particular engineering mindset. A failure in the initial construction can be catastrophic, and changing the project after the fact can be exceedingly difficult.
To ensure success and to avoid the challenges above smart contracts should here to best practices at their conception. Below, we summarized a checklist of key points & vulnerability vectors that help to indicate a high overall quality of the current Fetch.Ai project. (✓ indicates satisfaction; × indicates unsatisfaction; − indicates inapplicablility)

General

Compiling

- ✓ Correct environment settings, e.g. compiler version, test framework
- ✓ No compiler warnings

Logging

- ✓ Provide error message along with `assert` & `require`
- ✓ Use events to monitor contract activities

Code Layout

- ✓ According to Solidity Tutorial, Layout contract elements should following below order:
  1. Pragma statements
  2. Import statements
  3. Interfaces
  4. Libraries
  5. Contracts
- ✓ Each contract, library or interface should following below order:
  1. Type declarations
  2. State variables
  3. Events
  4. Functions
- ✓ According to Solidity Tutorial, functions should be grouped according to their visibility and ordered:
  1. constructor
  2. fallback function (if exists)
  3. external
  4. public
  5. internal
  6. private

Arithmetic Vulnerability

EVM specifies fixed-size data types for integers, in which means that has only a certain range of numbers it can store or represent.
Two's Complement / Integer underflow / overflow

- ✓ Use Math library as SafeMath for all arithmetic operations to handle integer overflow and underflow

Floating Points and Precision

- ✓ Correct handling the right precision when dealing ratios and rates

Access & Privilege Control Vulnerability

Circuit Breaker

- ✓ Provide pause functionality for control and emergency handling

Restriction

- ✓ Provide proper access control for functions
- ✓ Establish rate limiter for certain operations
- ✓ Restrict access to sensitive functions
- ✓ Restrict permission to contract destruction
- ✓ Establish speed bumps slow down some sensitive actions, any malicious actions occur, there is time to recover.

DoS Vulnerability

A type of attacks that make the contract inoperable with certain period of time or permanently.
Unexpected Revert

- ✓ Use favor pull over push pattern for handling unexpected revert

Block Gas Limit

- ✓ Use favor pull over push pattern for handling gas spent exceeds its limit on Contract via unbounded operations
- ✓ Use favor pull over push pattern for handling gas spent exceeds its limit on the network via block stuffing

Miner Manipulation Vulnerability

BlockNumber Dependence

- ✓ Understand the security risk level and trade-off of using `block.number` as one of core factors in the contract. Be aware that `block.number` can not be manipulated by the miner, but can lead to large than expected time differences. With assumptions of an Ethereum block confirmation takes 13 seconds. However, the average block time is between 13 15 seconds. During the difficulty bomb stage or hard/soft fork upgrade of the network, `block.number` to a time is dangerous and inaccurate as expected.

Timestamp Dependence

- ✓ Understand the security risk level and trade-off of using `block.timestamp` or alias `now` as one of core factors in the contract.
- ✓ Correct use of 15-second rule to minimize the impact caused by timestamp variance

Transaction Ordering Or Front-Running

- ✓ Understand the security risk level and the `gasPrice` rule in this vulnerability
- ✓ Correct placing an upper bound on the `gasPrice` for preventing the users taking the benefit of transaction ordering

External Referencing Vulnerability

External calls may execute malicious code in that contract or any other contract that it depends upon. As such, every external call should be treated as a potential security risk

- ✓ Correct using the pull over push favor for external calls to reduce reduces the chance of problems with the gas limit.

Avoid state changes after external calls

- ✓ Correct using checks-effects-interactions pattern to minimize the state changes after external contract or call referencing.

Handle errors in external calls

- ✓ Correct handling errors in any external contract or call referencing by checking its return value

Race Conditions Vulnerability

A type of vulnerability caused by calling external contracts that attacker can take over the control flow, and make changes to the data that the calling function wasn't expecting.

- Type of race conditions:
  - Reentrancy
    A state variable is changed after a contract uses `call.value()()`.

  - Cross-function Race Conditions

    An attacker may also be able to do a similar attack using two different functions that share the same state

- ✓ Avoid using `call.value()()`, instead use `send()`, `transfer()` that consumes 2300 gas. This will prevent any external code from being executed continuously
- ✓ Finish all internal work before calling the external function for unavoidable external call.

Low-level Call Vulnerability

The low-level function or opcodes are very useful and danger as for allowing the Libraries implementation and modularized code. However it opens up the doors to vulnerabilities as essentially your contract is allowing anyone to do whatever they want with their state
Code Injection by delegatecall

- ✓ Ensure the libraries implementation is stateless and non-self-destructable

Visibility Vulnerability

Solidity functions have 4 difference visibility dictate how functions are allowed to be called. The visibility determines whether a function can be called externally by users, by other derived contracts, only internally or only externally.

- ✓ Specify the visibility of all functions in a contract, even if they are intentionally public

Incorrect Interface Vulnerability

A contract interface defines functions with a different type signature than the implementation, causing two different method id's to be created. As a result, when the interface is called, the fallback method will be executed.

- ✓ Ensure the defined function signatures are match with the contract interface and implementation

Bad Randomness Vulnerability

Pseudo random number generation is not supported by Solidity as default, which it is an unsafe operation.

- ✓ Avoid using randomness for block variables, there may be a chance manipulated by the miners

Documentation

- ✓ Provide project README and execution guidance
- ✓ Provide inline comment for complex functions intention
- ✓ Provide instruction to initialize and execute the test files

Testing

- ✓ Provide migration scripts for easy contracts deployment to the Ethereum network
- ✓ Provide test scripts and coverage for potential scenarios

Overall we found the smart contracts to follow good practices. With the final update of source code and delivery of the audit report, we conclude that the contract is structurally sound and not vulnerable to any classically known anti-patterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend to seek multiple opinions, keep improving the codebase, and more test coverage and sandbox deployments before the mainnet release.

Static Analysis Results

TIMESTAMP_DEPENDENCY

Line 596 in File NativeTokenMigration.sol

```
596         require(block.timestamp >= _earliestDelete, "earliestDelete not reached");
```

”block.timestamp” can be influenced by miners to some degree

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address

Verification date	20, Oct 2018
Verification timespan	395.38 ms

CERTIK *label location*: Line 30-34 in File howtoread.sol

CERTIK *label*

```
30    /*@CTK FAIL "transferFrom to same address"
31        @tag assume_completion
32        @pre from == to
33        @post __post.allowed[from][msg.sender] ==
34    */
```

Raw code location: Line 35-41 in File howtoread.sol

Raw code

```
35    function transferFrom(address from, address to
          ) {
36        balances[from] = balances[from].sub(tokens
37        allowed[from][msg.sender] = allowed[from][
38        balances[to] = balances[to].add(tokens);
39        emit Transfer(from, to, tokens);
40        return true;
41    }
```

Counterexample: This code violates the specification

Initial environment

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          from = 0x0
5          to = 0x0
6          tokens = 0x6c
7      }
8      This = 0
53             balance: 0x0
54         }
55     }
```

Post environment

```
56
57 After Execution:
58     Input = {
59         from = 0x0
60         to = 0x0
61         tokens = 0x6c
```

Formal Verification Request 1

NativeTokenMigrationConstructor

13, Dec 2019
51.0 ms

Line 157-163 in File NativeTokenMigration.sol

```
157   /*@CTK NativeTokenMigrationConstructor
158     @post __post._upperTransferLimit == FET_TOTAL_SUPPLY
159     @post __post._lowerTransferLimit == 0
160     @post __post._transferFee == 0
161     @post __post._pausedTransferToNativeTargetAddress == false
162     @post __post._pausedTransferToNativeFromKey == false
163   */
```

Line 164-171 in File NativeTokenMigration.sol

```
164   constructor(address ERC20Address) public {
165       token = ERC20(ERC20Address);
166       _upperTransferLimit = FET_TOTAL_SUPPLY;
167       _lowerTransferLimit = 0;
168       _transferFee = 0;
169       _pausedTransferToNativeTargetAddress = false;
170       _pausedTransferToNativeFromKey = false;
171   }
```

The code meets the specification.

Formal Verification Request 2

_toNativeFET

13, Dec 2019
73.41 ms

Line 176-179 in File NativeTokenMigration.sol

```
176   /*@CTK _toNativeFET
177     @tag assume_completion
178     @post !__reverted -> __return == amount - amount % DECIMAL_DIFFERENTIATOR
179    */
```

Line 180-186 in File NativeTokenMigration.sol

```
180   function _toNativeFET(uint256 amount)
181   internal
182   pure
183   returns (uint256)
184   {
185       return amount.sub(amount.mod(DECIMAL_DIFFERENTIATOR));
186   }
```

The code meets the specification.

Formal Verification Request 3

_initSwap

13, Dec 2019
389.56 ms

Line 191-205 in File NativeTokenMigration.sol

```
191   /*@CTK "_initSwap"
192     @tag assume_completion
193     @pre amount <= _upperTransferLimit
194     @pre amount >= _lowerTransferLimit
195     @pre amount >= _transferFee
196     @post __post.globalSwapID == globalSwapID + 1
197
198     @post __post.swaps[globalSwapID ].sender == msg.sender
199     @post __post.swaps[globalSwapID ].amount == (amount - _transferFee) - ((amount -
            _transferFee) % DECIMAL_DIFFERENTIATOR)
200     @post __post.swaps[globalSwapID ].status == Status.Initialised
201     @post __post.swaps[globalSwapID ].fee == _transferFee
202
203     @post __post._completedAmount == _completedAmount + _transferFee + (amount -
            _transferFee) - ((amount - _transferFee) % DECIMAL_DIFFERENTIATOR)
204     @post __post._earliestDelete == block.timestamp + DELETE_PERIOD
205   */
```

Line 206-230 in File NativeTokenMigration.sol

```
206   function _initSwap(uint256 amount, string memory FETAddress)
207   internal
208   belowEqualValue(amount, _upperTransferLimit)
209   aboveEqualValue(amount, _lowerTransferLimit)
210   aboveEqualValue(amount, _transferFee)
211   {
212       uint256 id = globalSwapID;
213       globalSwapID = globalSwapID.add(1);
214
215       uint256 amountInt = _toNativeFET(amount.sub(_transferFee));
216
217       swaps[id].sender = msg.sender;
218       swaps[id].amount = amountInt;
219       swaps[id].status = Status.Initialised;
220       swaps[id].fee = _transferFee;
221
222       _completedAmount = _completedAmount.add(amountInt).add(_transferFee);
223       _earliestDelete = block.timestamp.add(DELETE_PERIOD);
224
225       require(token.transferFrom(msg.sender, address(this), amountInt.add(_transferFee)));
226
227       emit SwapInitialised(msg.sender, id, FETAddress, amountInt, _transferFee);
228   }
```

The code meets the specification.

Formal Verification Request 4

transferToNativeTargetAddress

13, Dec 2019
429.68 ms

Line 240-256 in File NativeTokenMigration.sol

```
240   /*@CTK transferToNativeTargetAddress
241     @pre !_pausedTransferToNativeTargetAddress
242
243     @tag assume_completion
244     @pre amount <= _upperTransferLimit
245     @pre amount >= _lowerTransferLimit
246     @pre amount >= _transferFee
247     @post __post.globalSwapID == globalSwapID + 1
248
249     @post __post.swaps[globalSwapID ].sender == msg.sender
250     @post __post.swaps[globalSwapID ].amount == (amount - _transferFee) - ((amount -
          _transferFee) % DECIMAL_DIFFERENTIATOR)
251     @post __post.swaps[globalSwapID ].status == Status.Initialised
252     @post __post.swaps[globalSwapID ].fee == _transferFee
253
254     @post __post._completedAmount == _completedAmount + _transferFee + (amount -
          _transferFee) - ((amount - _transferFee) % DECIMAL_DIFFERENTIATOR)
255     @post __post._earliestDelete == block.timestamp + DELETE_PERIOD
256   */
```

Line 257-263 in File NativeTokenMigration.sol

```
257   function transferToNativeTargetAddress(uint256 amount, string calldata FETAddress)
258   external
259   isFetchAddress(FETAddress)
260   whenNotPaused(_pausedTransferToNativeTargetAddress)
261   {
262       _initSwap(amount, FETAddress);
263   }
```

The code meets the specification.

Formal Verification Request 5

transferToNativeTargetAddress

13, Dec 2019
348.38 ms

Line 273-289 in File NativeTokenMigration.sol

```
273   /*@CTK transferToNativeTargetAddress
274     @pre !_pausedTransferToNativeFromKey
275
276     @tag assume_completion
277     @pre amount <= _upperTransferLimit
278     @pre amount >= _lowerTransferLimit
279     @pre amount >= _transferFee
280     @post __post.globalSwapID == globalSwapID + 1
```

```
281
282     @post __post.swaps[globalSwapID ].sender == msg.sender
283     @post __post.swaps[globalSwapID ].amount == (amount - _transferFee) - ((amount -
            _transferFee) % DECIMAL_DIFFERENTIATOR)
284     @post __post.swaps[globalSwapID ].status == Status.Initialised
285     @post __post.swaps[globalSwapID ].fee == _transferFee
286
287     @post __post._completedAmount == _completedAmount + _transferFee + (amount -
            _transferFee) - ((amount - _transferFee) % DECIMAL_DIFFERENTIATOR)
288     @post __post._earliestDelete == block.timestamp + DELETE_PERIOD
289   */
```

Line 290-295 in File NativeTokenMigration.sol

```
290   function transferToNativeFromKey(uint256 amount)
291   external
292   whenNotPaused(_pausedTransferToNativeFromKey)
293   {
294       _initSwap(amount, "");
295   }
```

✅ The code meets the specification.

Formal Verification Request 6

refund

📅 13, Dec 2019
⏱ 497.5 ms

Line 301-311 in File NativeTokenMigration.sol

```
301   /*@CTK refund
302     @tag assume_completion
303     @pre swaps[id].status == Status.Rejected
304     @pre swaps[id].sender == msg.sender
305
306     @post (swaps[id].amount + swaps[id].fee < swaps[id].amount || swaps[id].amount + swaps
            [id].fee < swaps[id].fee) -> __has_overflow
307     @post __post.swaps[id].sender == 0x0
308     @post __post.swaps[id].amount == 0
309     @post __post.swaps[id].status == Status.Empty
310     @post __post.swaps[id].fee == 0
311    */
```

Line 312-321 in File NativeTokenMigration.sol

```
312   function refund(uint256 id)
313   external
314   isRejected(id)
315   onlySender(id)
316   {
317       uint256 amount = swaps[id].amount.add(swaps[id].fee);
318       emit Refund(msg.sender, id);
319       delete swaps[id];
320       require(token.transfer(msg.sender, amount));
321   }
```

✅ The code meets the specification.

Formal Verification Request 7

requestRefund

13, Dec 2019
33.75 ms

Line 328-331 in File NativeTokenMigration.sol

```
328     /*@CTK requestRefund
329       @pre swaps[id].status == Status.Initialised
330       @pre swaps[id].sender == msg.sender
331      */
```

Line 332-338 in File NativeTokenMigration.sol

```
332     function requestRefund(uint256 id)
333     external
334     isInitialised(id)
335     onlySender(id)
336     {
337         emit RefundRequested(msg.sender, id, swaps[id].amount);
338     }
```

The code meets the specification.

Formal Verification Request 8

pauseTransferToNativeTargetAddress

13, Dec 2019
53.29 ms

Line 348-352 in File NativeTokenMigration.sol

```
348     /*@CTK pauseTransferToNativeTargetAddress
349       @pre msg.sender == _owner || delegates[msg.sender]
350
351       @post __post._pausedTransferToNativeTargetAddress == isPaused
352      */
```

Line 353-359 in File NativeTokenMigration.sol

```
353     function pauseTransferToNativeTargetAddress(bool isPaused)
354     external
355     onlyDelegate()
356     {
357         _pausedTransferToNativeTargetAddress = isPaused;
358         emit PauseTransferToNativeTargetAddress(isPaused);
359     }
```

The code meets the specification.

Formal Verification Request 9

pauseTransferToNativeFromKey

13, Dec 2019
48.24 ms

Line 366-369 in File NativeTokenMigration.sol

```
366     /*@CTK pauseTransferToNativeFromKey
367       @pre msg.sender == _owner || delegates[msg.sender]
368       @post __post._pausedTransferToNativeFromKey == isPaused
369      */
```

Line 370-376 in File NativeTokenMigration.sol

```
370     function pauseTransferToNativeFromKey(bool isPaused)
371     external
372     onlyDelegate()
373     {
374         _pausedTransferToNativeFromKey = isPaused;
375         emit PauseTransferToNativeFromKey(isPaused);
376     }
```

✅ The code meets the specification.

Formal Verification Request 10

setDelegate

📅 13, Dec 2019
⏱ 48.74 ms

Line 384-387 in File NativeTokenMigration.sol

```
384     /*@CTK setDelegate
385       @pre msg.sender == _owner
386       @post __post.delegates[_address] == isDelegate
387      */
```

Line 388-394 in File NativeTokenMigration.sol

```
388     function setDelegate(address _address, bool isDelegate)
389     external
390     onlyOwner()
391     {
392         delegates[_address] = isDelegate;
393         emit ChangeDelegate(_address, isDelegate);
394     }
```

✅ The code meets the specification.

Formal Verification Request 11

setUpperTransferLimit

📅 13, Dec 2019
⏱ 83.96 ms

Line 401-406 in File NativeTokenMigration.sol

```
401     /*@CTK setUpperTransferLimit
402       @pre msg.sender == _owner
403       @pre newLimit <= FET_TOTAL_SUPPLY
404       @pre newLimit >= _lowerTransferLimit
405       @post __post._upperTransferLimit == newLimit
406      */
```

Line 407-415 in File NativeTokenMigration.sol

```
407     function setUpperTransferLimit(uint256 newLimit)
408     external
409     onlyOwner()
410     belowEqualValue(newLimit, FET_TOTAL_SUPPLY)
411     aboveEqualValue(newLimit, _lowerTransferLimit)
412     {
413         _upperTransferLimit = newLimit;
414         emit ChangeUpperTransferLimit(newLimit);
415     }
```

✅ The code meets the specification.

Formal Verification Request 12

setLowerTransferLimit

📅 13, Dec 2019
⏱ 66.34 ms

Line 422-426 in File NativeTokenMigration.sol

```
422     /*@CTK setLowerTransferLimit
423       @pre msg.sender == _owner
424       @pre newLimit <= _upperTransferLimit
425       @post __post._lowerTransferLimit == newLimit
426      */
```

Line 427-434 in File NativeTokenMigration.sol

```
427     function setLowerTransferLimit(uint256 newLimit)
428     external
429     onlyOwner()
430     belowEqualValue(newLimit, _upperTransferLimit)
431     {
432         _lowerTransferLimit = newLimit;
433         emit ChangeLowerTransferLimit(newLimit);
434     }
```

✅ The code meets the specification.

Formal Verification Request 13

setTransferFee

📅 13, Dec 2019
⏱ 47.72 ms

Line 442-445 in File NativeTokenMigration.sol

```
442     /*@CTK setTransferFee
443       @pre msg.sender == _owner
444       @post __post._transferFee == newFee
445      */
```

Line 446-452 in File NativeTokenMigration.sol

```
446   function setTransferFee(uint256 newFee)
447   external
448   onlyOwner()
449   {
450       _transferFee = newFee;
451       emit ChangeTransferFee(newFee);
452   }
```

The code meets the specification.

Formal Verification Request 14

_reject

13, Dec 2019
241.94 ms

Line 454-460 in File NativeTokenMigration.sol

```
454   /*@CTK _reject
455     @tag assume_completion
456     @pre swaps[id].status == Status.Initialised
457     @pre block.number <= expirationBlock
458     @post __post.swaps[id].status == Status.Rejected
459     @post __post._completedAmount == _completedAmount - swaps[id].amount - swaps[id].fee
460    */
```

Line 461-469 in File NativeTokenMigration.sol

```
461   function _reject(address sender, uint256 id, uint256 expirationBlock, string memory
          reason)
462   internal
463   isInitialised(id)
464   belowEqualValue(block.number, expirationBlock)
465   {
466       emit Rejected(sender, id, reason);
467       swaps[id].status = Status.Rejected;
468       _completedAmount = _completedAmount.sub(swaps[id].amount).sub(swaps[id].fee);
469   }
```

The code meets the specification.

Formal Verification Request 15

reject

13, Dec 2019
489.34 ms

Line 478-484 in File NativeTokenMigration.sol

```
478     /*@CTK reject
479       @tag assume_completion
480       @pre msg.sender == _owner || delegates[msg.sender] @pre swaps[id].status == Status.
              Initialised
481       @pre block.number <= expirationBlock
482       @post __post.swaps[id].status == Status.Rejected
483       @post __post._completedAmount == _completedAmount - (swaps[id].amount + swaps[id].fee)
484      */
```

Line 485-490 in File NativeTokenMigration.sol

```
485     function reject(address sender, uint256 id, uint256 expirationBlock, string calldata
            reason)
486     external
487     onlyDelegate()
488     {
489         _reject(sender, id, expirationBlock, reason);
490     }
```

The code meets the specification.

Formal Verification Request 16

batchReject

13, Dec 2019
74.03 ms

Line 500-506 in File NativeTokenMigration.sol

```
500     /*@CTK batchReject
501       @tag assume_completion
502       @pre msg.sender == _owner || delegates[msg.sender]
503       @pre senders.length == _ids.length
504       @pre senders.length == expirationBlocks.length
505       @post !__reverted
506      */
```

Line 507-525 in File NativeTokenMigration.sol

```
507     function batchReject(address[] calldata senders,
508         uint256[] calldata _ids,
509         uint256[] calldata expirationBlocks,
510         string calldata reason)
511     external
512     onlyDelegate()
513     isEqual(senders.length, _ids.length)
514     isEqual(senders.length, expirationBlocks.length)
515     {
516         /*@CTK batchReject_loop
517           @inv i <= senders.length
518           @inv forall j: uint. (j >= 0 /\ j < i) -> this.swaps[_ids[j]].status == Status.
                  Rejected
519           @post i == senders.length
520           @post !__should_return
521         */
522         for (uint256 i = 0; i < senders.length; i++) {
523             _reject(senders[i], _ids[i], expirationBlocks[i], reason);
```

```
524         }
525     }
```

The code meets the specification.

Formal Verification Request 17

withdrawToFoundation

13, Dec 2019
167.66 ms

Line 536-543 in File NativeTokenMigration.sol

```
536     /*@CTK withdrawToFoundation
537       @tag assume_completion
538       @pre msg.sender == _owner
539       @pre _amount <= _completedAmount
540       @post _amount == 0 -> __post.amount == _completedAmount
541       @post _amount != 0 -> __post.amount == _amount
542       @post __post._completedAmount == _completedAmount - __post.amount
543      */
```

Line 544-562 in File NativeTokenMigration.sol

```
544     function withdrawToFoundation(uint256 _amount)
545     external
546     onlyOwner()
547     belowEqualValue(_amount, _completedAmount)
548     {
549         uint amount = 0;
550 
551         if (_amount == 0) {
552             amount = _completedAmount;
553         } else {
554             amount = _amount;
555         }
556         _completedAmount = _completedAmount.sub(amount);
557         require(token.transfer(owner(), amount));
558         emit WithdrawalToFoundation(amount);
559     }
```

The code meets the specification.

Formal Verification Request 18

topupCompletedAmount

13, Dec 2019
36.56 ms

Line 569-572 in File NativeTokenMigration.sol

```
569     /*@CTK topupCompletedAmount
570       @tag assume_completion
571       @pre __post._completedAmount == _completedAmount + amount
572      */
```

Line 573-580 in File NativeTokenMigration.sol

```
573     function topupCompletedAmount(uint256 amount)
574     external
575     {
576         _completedAmount = _completedAmount.add(amount);
577         require(token.transferFrom(msg.sender, address(this), amount));
578     }
```

The code meets the specification.

Formal Verification Request 19

deleteContract

13, Dec 2019
44.89 ms

Line 588-591 in File NativeTokenMigration.sol

```
588     /*@CTK deleteContract
589      @pre msg.sender == _owner
590      @pre block.timestamp >= _earliestDelete
591      */
```

Line 592-603 in File NativeTokenMigration.sol

```
592     function deleteContract(address payable payoutAddress)
593     external
594     onlyOwner()
595     {
596         require(block.timestamp >= _earliestDelete, "earliestDelete not reached");
597         uint256 contractBalance = token.balanceOf(address(this));
598         require(token.transfer(payoutAddress, contractBalance));
599         emit DeleteContract();
600         selfdestruct(payoutAddress);
601     }
```

The code meets the specification.

Formal Verification Request 20

batchReject__loop___Generated

13, Dec 2019
205.28 ms

(Loop) Line 516-521 in File NativeTokenMigration.sol

```
516         /*@CTK batchReject_loop
517           @inv i <= senders.length
518           @inv forall j: uint. (j >= 0 /\ j < i) -> this.swaps[_ids[j]].status == Status.
                  Rejected
519           @post i == senders.length
520           @post !__should_return
521         */
```

(Loop) Line 516-524 in File NativeTokenMigration.sol

```
516         /*@CTK batchReject_loop
517           @inv i <= senders.length
518           @inv forall j: uint. (j >= 0 /\ j < i) -> this.swaps[_ids[j]].status == Status.
                  Rejected
519           @post i == senders.length
520           @post !__should_return
521         */
522         for (uint256 i = 0; i < senders.length; i++) {
523             _reject(senders[i], _ids[i], expirationBlocks[i], reason);
524         }
```

The code meets the specification.

Source Code with CertiK Labels

File NativeTokenMigration.sol

```
1  pragma solidity ^0.5.13;
2
3  /* import "openzeppelin-solidity/contracts/token/ERC20/ERC20.sol"; */
4  import "../contracts_openzeppelin/ERC20/ERC20.sol";
5  /* import "openzeppelin-solidity/contracts/lifecycle/Pausable.sol"; */
6  /* import 'openzeppelin-solidity/contracts/ownership/Ownable.sol'; */
7  import "../contracts_openzeppelin/Ownable.sol";
8  //import "../contracts_openzeppelin/SafeMath.sol"; // Imported by ERC20.sol
9
10
11 interface NativeTokenMigrationInterface {
12     // ***
13     // Public functions
14     // ***
15     function transferToNativeTargetAddress(uint256 amount, string calldata FETAddress)
           external;
16     function transferToNativeFromKey(uint256 amount) external;
17     function refund(uint256 id) external;
18     function requestRefund(uint256 id) external;
19
20     // ***
21     // Restricted functions: Owner only
22     // ***
23     // Add or remove a delegate address that is allowed to confirm and reject transactions
24     function setDelegate(address _address, bool isDelegate) external;
25     // Change the _upperTransferLimit which is the maximum threshold any single swap can be
26     function setUpperTransferLimit(uint256 newLimit) external;
27     // Change the _lowerTransferLimit which is the minimum threshold any single swap can be
28     function setLowerTransferLimit(uint256 newLimit) external;
29     // Withdraw the tokens the confirmed swaps to the owner
30     function withdrawToFoundation(uint256 _amount) external;
31     // Delete the contract after _earliestDelete timestamp is reached, transfers the
           remaining
32     function deleteContract(address payable payoutAddress) external;
33
34     // ***
35     // Restricted functions: Owner or delegate only
36     // ***
37     // Reject a swap with reason. Allows the initialiser to immediately withdraw the funds
           again
38     function reject(address sender, uint256 id, uint256 expirationBlock, string calldata
           reason) external;
39     // Reject multiple swaps with the same reason
40     function batchReject(
41         address[] calldata senders,
42         uint256[] calldata _ids,
43         uint256[] calldata expirationBlocks,
44         string calldata reason) external;
45     // Pause or unpause the transferToNativeTargetAddress() method
46     function pauseTransferToNativeTargetAddress(bool isPaused) external;
47     // Pause or unpause the transferToNativeFromKey() method
48     function pauseTransferToNativeFromKey(bool isPaused) external;
49 }
50
```

```
51
52  contract NativeTokenMigration is Ownable, NativeTokenMigrationInterface {
53      using SafeMath for uint256;
54
55      // minimum time the owner has to wait after the last initialised transfer before allowed
            to
56      // delete the contract, in seconds
57      uint256 constant DELETE_PERIOD = 10 days;
58      uint256 constant FET_TOTAL_SUPPLY = 1152997575 * 10**18;
59      uint256 constant DECIMAL_DIFFERENTIATOR = 10**8;
60
61      enum Status {Empty, Initialised, Rejected}
62
63      struct Swap {
64          address sender;
65          uint256 amount;
66          Status status;
67          uint256 fee;
68      }
69
70      uint256 public globalSwapID;
71      // globalSwapID => swap
72      // usage of a global id instead of address, id pair to simplify processing during the
            transfer
73      // over to mainnet
74      mapping(uint256 => Swap) public swaps;
75      // global counter increased by every new swap
76
77      ERC20 public token;
78
79      mapping(address => bool) public delegates;
80      uint256 public _transferFee;
81      uint256 public _upperTransferLimit;
82      uint256 public _lowerTransferLimit;
83      uint256 public _completedAmount;
84      bool public _pausedTransferToNativeTargetAddress;
85      bool public _pausedTransferToNativeFromKey;
86      uint256 public _earliestDelete;
87
88      modifier belowEqualValue(uint256 amount, uint256 threshold) {
89          require(amount <= threshold, "Value too high");
90          _;
91      }
92
93      modifier aboveEqualValue(uint256 amount, uint256 threshold) {
94          require(amount >= threshold, "Value too low");
95          _;
96      }
97
98      /* Simple length check. Length of FET addresses seem to be either 49 or
99          50 bytes. Adding a slight margin to this. A proper checksum validation would require
                a base58
100         decoder.*/
101     modifier isFetchAddress(string memory _address) {
102         require(bytes(_address).length > 47, "Address too short");
103         require(bytes(_address).length < 52, "Address too long");
104         _;
105     }
```

```
106
107     modifier onlySender(uint256 id) {
108         require(swaps[id].sender == msg.sender, "Not the sender");
109         _;
110     }
111
112     /* Only callable by owner or delegate */
113     modifier onlyDelegate() {
114         require(isOwner() || delegates[msg.sender], "Caller is neither owner nor delegate");
115         _;
116     }
117
118     modifier isEqual(uint256 a, uint256 b) {
119         require(a == b, "Different values");
120         _;
121     }
122
123     modifier whenNotPaused(bool pauseIndicator) {
124         require(!pauseIndicator, "Transfers are paused");
125         _;
126     }
127
128     modifier isRejected(uint256 id) {
129         require(swaps[id].status == Status.Rejected, "The swap has not been rejected");
130         _;
131     }
132
133     modifier isInitialised(uint256 id){
134         require(swaps[id].status == Status.Initialised, "The swap has not been initialised");

135         _;
136     }
137
138     event SwapInitialised(address indexed sender, uint256 indexed id, string FETAddress,
        uint256 amount, uint256 fee);
139     event Rejected(address indexed sender, uint256 indexed id, string reason);
140     event Refund(address indexed sender, uint256 indexed id);
141     event RefundRequested(address indexed sender, uint256 indexed id, uint256 amount);
142     event PauseTransferToNativeTargetAddress(bool isPaused);
143     event PauseTransferToNativeFromKey(bool isPaused);
144     event ChangeDelegate(address delegate, bool isDelegate);
145     event ChangeUpperTransferLimit(uint256 newLimit);
146     event ChangeLowerTransferLimit(uint256 newLimit);
147     event ChangeTransferFee(uint256 newFee);
148     event DeleteContract();
149     event WithdrawalToFoundation(uint256 amount);
150
151     /*******************
152     Contract start
153     *******************/
154     /**
155      * @param ERC20Address address of the ERC20 contract
156      */
157     /*@CTK NativeTokenMigrationConstructor
158       @post __post._upperTransferLimit == FET_TOTAL_SUPPLY
159       @post __post._lowerTransferLimit == 0
160       @post __post._transferFee == 0
161       @post __post._pausedTransferToNativeTargetAddress == false
```

```
162       @post __post._pausedTransferToNativeFromKey == false
163     */
164     constructor(address ERC20Address) public {
165         token = ERC20(ERC20Address);
166         _upperTransferLimit = FET_TOTAL_SUPPLY;
167         _lowerTransferLimit = 0;
168         _transferFee = 0;
169         _pausedTransferToNativeTargetAddress = false;
170         _pausedTransferToNativeFromKey = false;
171     }
172
173     /**
174      * @notice Return a unit that is divisible by the Fetch mainnet precision
175      */
176     /*@CTK _toNativeFET
177       @tag assume_completion
178       @post !__reverted -> __return == amount - amount % DECIMAL_DIFFERENTIATOR
179      */
180     function _toNativeFET(uint256 amount)
181     internal
182     pure
183     returns (uint256)
184     {
185         return amount.sub(amount.mod(DECIMAL_DIFFERENTIATOR));
186     }
187
188     /**
189      * @notice Initialise a swap. Internal only.
190      */
191     /*@CTK "_initSwap"
192       @tag assume_completion
193       @pre amount <= _upperTransferLimit
194       @pre amount >= _lowerTransferLimit
195       @pre amount >= _transferFee
196       @post __post.globalSwapID == globalSwapID + 1
197
198       @post __post.swaps[globalSwapID ].sender == msg.sender
199       @post __post.swaps[globalSwapID ].amount == (amount - _transferFee) - ((amount -
              _transferFee) % DECIMAL_DIFFERENTIATOR)
200       @post __post.swaps[globalSwapID ].status == Status.Initialised
201       @post __post.swaps[globalSwapID ].fee == _transferFee
202
203       @post __post._completedAmount == _completedAmount + _transferFee + (amount -
              _transferFee) - ((amount - _transferFee) % DECIMAL_DIFFERENTIATOR)
204       @post __post._earliestDelete == block.timestamp + DELETE_PERIOD
205     */
206     function _initSwap(uint256 amount, string memory FETAddress)
207     internal
208     belowEqualValue(amount, _upperTransferLimit)
209     aboveEqualValue(amount, _lowerTransferLimit)
210     aboveEqualValue(amount, _transferFee)
211     {
212         uint256 id = globalSwapID;
213         globalSwapID = globalSwapID.add(1);
214
215         uint256 amountInt = _toNativeFET(amount.sub(_transferFee));
216
217         swaps[id].sender = msg.sender;
```

```
218         swaps[id].amount = amountInt;
219         swaps[id].status = Status.Initialised;
220         swaps[id].fee = _transferFee;
221
222         _completedAmount = _completedAmount.add(amountInt).add(_transferFee);
223         _earliestDelete = block.timestamp.add(DELETE_PERIOD);
224
225         require(token.transferFrom(msg.sender, address(this), amountInt.add(_transferFee)));
226
227         emit SwapInitialised(msg.sender, id, FETAddress, amountInt, _transferFee);
228     }
229
230     /**
231      * @notice Initialise a swap to an address on the Fetch mainnet
232      * @param amount amount to transfer. Must be below _upperTransferLimit
233      * @param FETAddress public target address on the Fetch mainnet to transfer the tokens
            to
234      * @dev Disregards fractions of FET due to precision differences
235      * @dev The transfer of ERC20 tokens requires to first approve this transfer with the
            ERC20
236        contract by calling ERC20.approve(contractAddress, amount)
237      */
238     /*@CTK transferToNativeTargetAddress
239       @pre !_pausedTransferToNativeTargetAddress
240
241       @tag assume_completion
242       @pre amount <= _upperTransferLimit
243       @pre amount >= _lowerTransferLimit
244       @pre amount >= _transferFee
245       @post __post.globalSwapID == globalSwapID + 1
246
247       @post __post.swaps[globalSwapID ].sender == msg.sender
248       @post __post.swaps[globalSwapID ].amount == (amount - _transferFee) - ((amount -
            _transferFee) % DECIMAL_DIFFERENTIATOR)
249       @post __post.swaps[globalSwapID ].status == Status.Initialised
250       @post __post.swaps[globalSwapID ].fee == _transferFee
251
252       @post __post._completedAmount == _completedAmount + _transferFee + (amount -
            _transferFee) - ((amount - _transferFee) % DECIMAL_DIFFERENTIATOR)
253       @post __post._earliestDelete == block.timestamp + DELETE_PERIOD
254     */
255     function transferToNativeTargetAddress(uint256 amount, string calldata FETAddress)
256     external
257     isFetchAddress(FETAddress)
258     whenNotPaused(_pausedTransferToNativeTargetAddress)
259     {
260         _initSwap(amount, FETAddress);
261     }
262
263     /**
264      * @notice Initialise a swap to an address on the Fetch mainnet that corresponds to the
            same
265        private key as used to control the address invoking this address
266      * @param amount amount to transfer. Must be below _upperTransferLimit
267      * @dev Disregards fractions of FET due to precision differences
268      * @dev The transfer of ERC20 tokens requires to first approve this transfer with the
            ERC20
269        contract by calling ERC20.approve(contractAddress, amount)
```

```
270      */
271     /*@CTK transferToNativeTargetAddress
272       @pre !_pausedTransferToNativeFromKey
273
274       @tag assume_completion
275       @pre amount <= _upperTransferLimit
276       @pre amount >= _lowerTransferLimit
277       @pre amount >= _transferFee
278       @post __post.globalSwapID == globalSwapID + 1
279
280       @post __post.swaps[globalSwapID ].sender == msg.sender
281       @post __post.swaps[globalSwapID ].amount == (amount - _transferFee) - ((amount -
              _transferFee) % DECIMAL_DIFFERENTIATOR)
282       @post __post.swaps[globalSwapID ].status == Status.Initialised
283       @post __post.swaps[globalSwapID ].fee == _transferFee
284
285       @post __post._completedAmount == _completedAmount + _transferFee + (amount -
              _transferFee) - ((amount - _transferFee) % DECIMAL_DIFFERENTIATOR)
286       @post __post._earliestDelete == block.timestamp + DELETE_PERIOD
287     */
288     function transferToNativeFromKey(uint256 amount)
289     external
290     whenNotPaused(_pausedTransferToNativeFromKey)
291     {
292         _initSwap(amount, "");
293     }
294
295     /**
296      * @notice Reclaim tokens of a swap that has been rejected
297      * @param id id of the swap to refund
298      */
299     /*@CTK refund
300       @tag assume_completion
301       @pre swaps[id].status == Status.Rejected
302       @pre swaps[id].sender == msg.sender
303
304       @post (swaps[id].amount + swaps[id].fee < swaps[id].amount || swaps[id].amount + swaps
              [id].fee < swaps[id].fee) -> __has_overflow
305       @post __post.swaps[id].sender == 0x0
306       @post __post.swaps[id].amount == 0
307       @post __post.swaps[id].status == Status.Empty
308       @post __post.swaps[id].fee == 0
309      */
310     function refund(uint256 id)
311     external
312     isRejected(id)
313     onlySender(id)
314     {
315         uint256 amount = swaps[id].amount.add(swaps[id].fee);
316         emit Refund(msg.sender, id);
317         delete swaps[id];
318         require(token.transfer(msg.sender, amount));
319     }
320
321     /**
322      * @notice Request that a refund be issued. Allows users to "complain" and remind the
              automated
323      * server that it might have missed an event somewhere and should try reprocessing it
```

```
324      * @param id id of the swap to refund
325      */
326     /*@CTK requestRefund
327       @pre swaps[id].status == Status.Initialised
328       @pre swaps[id].sender == msg.sender
329      */
330     function requestRefund(uint256 id)
331     external
332     isInitialised(id)
333     onlySender(id)
334     {
335         emit RefundRequested(msg.sender, id, swaps[id].amount);
336     }
337
338     /********************
339     Restricted functions
340     ********************/
341     /**
342      * @notice Pause or unpause the transferToNativeTargetAddress() method
343      * @param isPaused whether to pause or unpause the method
344      * @dev Delegate only
345      */
346     /*@CTK pauseTransferToNativeTargetAddress
347       @pre msg.sender == _owner || delegates[msg.sender]
348
349       @post __post._pausedTransferToNativeTargetAddress == isPaused
350      */
351     function pauseTransferToNativeTargetAddress(bool isPaused)
352     external
353     onlyDelegate()
354     {
355         _pausedTransferToNativeTargetAddress = isPaused;
356         emit PauseTransferToNativeTargetAddress(isPaused);
357     }
358
359     /**
360      * @notice Pause or unpause the transferToNativeFromKey() method
361      * @param isPaused whether to pause or unpause the method
362      * @dev Delegate only
363      */
364     /*@CTK pauseTransferToNativeFromKey
365       @pre msg.sender == _owner || delegates[msg.sender]
366       @post __post._pausedTransferToNativeFromKey == isPaused
367      */
368     function pauseTransferToNativeFromKey(bool isPaused)
369     external
370     onlyDelegate()
371     {
372         _pausedTransferToNativeFromKey = isPaused;
373         emit PauseTransferToNativeFromKey(isPaused);
374     }
375
376     /**
377      * @notice Add or remove a delegate address that is allowed to confirm and reject
          transactions
378      * @param _address address of the delegate
379      * @param isDelegate whether to add or remove the address from the delegates set
380      * @dev Owner only
```

```
381      */
382      /*@CTK setDelegate
383        @pre msg.sender == _owner
384        @post __post.delegates[_address] == isDelegate
385       */
386     function setDelegate(address _address, bool isDelegate)
387     external
388     onlyOwner()
389     {
390         delegates[_address] = isDelegate;
391         emit ChangeDelegate(_address, isDelegate);
392     }
393
394     /**
395      * @notice Change the _upperTransferLimit which is the maximum threshold any single swap
             can be
396      * @param newLimit new limit in FET * 10**18
397      * @dev Owner only
398      */
399      /*@CTK setUpperTransferLimit
400        @pre msg.sender == _owner
401        @pre newLimit <= FET_TOTAL_SUPPLY
402        @pre newLimit >= _lowerTransferLimit
403        @post __post._upperTransferLimit == newLimit
404       */
405     function setUpperTransferLimit(uint256 newLimit)
406     external
407     onlyOwner()
408     belowEqualValue(newLimit, FET_TOTAL_SUPPLY)
409     aboveEqualValue(newLimit, _lowerTransferLimit)
410     {
411         _upperTransferLimit = newLimit;
412         emit ChangeUpperTransferLimit(newLimit);
413     }
414
415     /**
416      * @notice Change the _lowerTransferLimit which is the minimum threshold any single swap
             can be
417      * @param newLimit new limit in FET * 10**18
418      * @dev Owner only
419      */
420      /*@CTK setLowerTransferLimit
421        @pre msg.sender == _owner
422        @pre newLimit <= _upperTransferLimit
423        @post __post._lowerTransferLimit == newLimit
424       */
425     function setLowerTransferLimit(uint256 newLimit)
426     external
427     onlyOwner()
428     belowEqualValue(newLimit, _upperTransferLimit)
429     {
430         _lowerTransferLimit = newLimit;
431         emit ChangeLowerTransferLimit(newLimit);
432     }
433
434     /**
435      * @notice Change the _transferFee which is the fee applied to every initialised swap
436      * @param newFee in FET * 10**18
```

```
437      * @dev This fee will be refunded if the swap is rejected
438      * @dev Owner only
439      */
440      /*@CTK setTransferFee
441        @pre msg.sender == _owner
442        @post __post._transferFee == newFee
443       */
444     function setTransferFee(uint256 newFee)
445     external
446     onlyOwner()
447     {
448         _transferFee = newFee;
449         emit ChangeTransferFee(newFee);
450     }
451
452     /*@CTK _reject
453       @tag assume_completion
454       @pre swaps[id].status == Status.Initialised
455       @pre block.number <= expirationBlock
456       @post __post.swaps[id].status == Status.Rejected
457       @post __post._completedAmount == _completedAmount - swaps[id].amount - swaps[id].fee
458      */
459     function _reject(address sender, uint256 id, uint256 expirationBlock, string memory
         reason)
460     internal
461     isInitialised(id)
462     belowEqualValue(block.number, expirationBlock)
463     {
464         emit Rejected(sender, id, reason);
465         swaps[id].status = Status.Rejected;
466         _completedAmount = _completedAmount.sub(swaps[id].amount).sub(swaps[id].fee);
467     }
468
469     /**
470      * @notice Reject a swap with reason. Allows the initialiser to immediately withdraw the
             funds again
471      * @param sender initialiser of the swap
472      * @param id id of the swap
473      * @param reason reason for rejection
474      * @dev delegate only
475      */
476     /*@CTK reject
477       @tag assume_completion
478       @pre msg.sender == _owner || delegates[msg.sender] @pre swaps[id].status == Status.
             Initialised
479       @pre block.number <= expirationBlock
480       @post __post.swaps[id].status == Status.Rejected
481       @post __post._completedAmount == _completedAmount - (swaps[id].amount + swaps[id].fee)
482      */
483     function reject(address sender, uint256 id, uint256 expirationBlock, string calldata
         reason)
484     external
485     onlyDelegate()
486     {
487         _reject(sender, id, expirationBlock, reason);
488     }
489
490     /**
```

```
491      * @notice Reject multiple swaps with the same reason
492      * @param senders array of sender addresses
493      * @param _ids array of swap id's
494      * @param reason Reason for the rejection. Will be identical across all swaps due to
             string[]
495        being only an experimental feature
496      * @dev delegate only
497      */
498     /*@CTK batchReject
499       @tag assume_completion
500       @pre msg.sender == _owner || delegates[msg.sender]
501       @pre senders.length == _ids.length
502       @pre senders.length == expirationBlocks.length
503       @post !__reverted
504      */
505     function batchReject(address[] calldata senders,
506         uint256[] calldata _ids,
507         uint256[] calldata expirationBlocks,
508         string calldata reason)
509     external
510     onlyDelegate()
511     isEqual(senders.length, _ids.length)
512     isEqual(senders.length, expirationBlocks.length)
513     {
514         /*@CTK batchReject_loop
515           @inv i <= senders.length
516           @inv forall j: uint. (j >= 0 /\ j < i) -> this.swaps[_ids[j]].status == Status.
                  Rejected
517           @post i == senders.length
518           @post !__should_return
519         */
520         for (uint256 i = 0; i < senders.length; i++) {
521             _reject(senders[i], _ids[i], expirationBlocks[i], reason);
522         }
523     }
524
525     /**
526      * @notice Withdraw the tokens the confirmed swaps to the owner
527      * @param _amount amount to withdraw. Set to zero to withdraw all.
528      * @dev owner only
529      */
530
531     /*@CTK withdrawToFoundation
532       @tag assume_completion
533       @pre msg.sender == _owner
534       @pre _amount <= _completedAmount
535       @post _amount == 0 -> __post.amount == _completedAmount
536       @post _amount != 0 -> __post.amount == _amount
537       @post __post._completedAmount == _completedAmount - __post.amount
538      */
539     function withdrawToFoundation(uint256 _amount)
540     external
541     onlyOwner()
542     belowEqualValue(_amount, _completedAmount)
543     {
544         uint amount = 0;
545
546         if (_amount == 0) {
```

```
547             amount = _completedAmount;
548         } else {
549             amount = _amount;
550         }
551         _completedAmount = _completedAmount.sub(amount);
552         require(token.transfer(owner(), amount));
553         emit WithdrawalToFoundation(amount);
554     }
555 
556     /**
557      * @notice Fallback function that allows to increase _completedAmount if the foundation
             should
558      *  ever withdraw more than required to refund rejected swaps
559      * @param amount amount to increase _completedAmount by
560      */
561     /*@CTK topupCompletedAmount
562       @tag assume_completion
563       @pre __post._completedAmount == _completedAmount + amount
564      */
565     function topupCompletedAmount(uint256 amount)
566     external
567     {
568         _completedAmount = _completedAmount.add(amount);
569         require(token.transferFrom(msg.sender, address(this), amount));
570     }
571 
572     /**
573      * @notice Delete the contract after _earliestDelete timestamp is reached, transfers the
             remaining
574         token and ether balance to the specified payoutAddress
575      * @param payoutAddress address to transfer the balances to. Ensure that this is able to
             handle ERC20 tokens
576      * @dev owner only
577      */
578     /*@CTK deleteContract
579       @pre msg.sender == _owner
580       @pre block.timestamp >= _earliestDelete
581      */
582     function deleteContract(address payable payoutAddress)
583     external
584     onlyOwner()
585     {
586         require(block.timestamp >= _earliestDelete, "earliestDelete not reached");
587         uint256 contractBalance = token.balanceOf(address(this));
588         require(token.transfer(payoutAddress, contractBalance));
589         emit DeleteContract();
590         selfdestruct(payoutAddress);
591     }
592 
593 }
```