A CONSENSYS DILIGENCE AUDIT REPORT

# Codefi ERC1400 Assessment

| | |
|---|---|
| **Date** | June 2020 |
| **Auditors** | Daniel Luca |

## 1 Executive Summary

This report presents the results of our engagement with **Codefi** to review **ERC-1400**.

The review was conducted over the course of two weeks, **from June 1 to June 12 2020**, by Daniel Luca. A total of **10 person-days** were spent.

During the **first week**, we reviewed the changes since the last code review our team did in June 2019 but quickly realized that it would not be effective since a lot of code was changed. We set up a meeting with the development team on Tuesday to explain our process, understand the changes, agree on the scope of the review, pick a commit hash and where we should focus mostly.

We agreed to audit the current version of `ERC1400.sol` and not focus on the changes since some do not reflect the existing code. The initial meeting with the development team revealed that we should focus on the contract extension, contract size optimization and gas optimizations; of course, any security issues should also be reported.

The EIP-1400 has suffered many changes, forcing the development team to keep the implementation in sync with the evolving standard.

We ran a few tools to check for glaring security problems and to create a better overview of how the implementation works. The output of these tools was added in the Artifacts section.

We proceeded to check the current implementation against the EIP-1400 standard. The code is well documented and well structured and it was easy to read and understand.

During the **second week**, we continued our code review and proceeded to check all of the remaining functionality. Again, no significant problems were found in the code.

We set up a couple of meetings with the development team to discuss decisions and initial findings. We presented ideas on how to reduce code complexity and how to improve gas consumption, return error codes and different caveats.

Towards the end of the week, the client added to the scope of the audit two additional contracts: `ERC1400CertificateNonce.sol` and `ERC1400CertificateSalt.sol`. Their code was audited previously in the previous review in June 2019, but the structure of the contracts was changed.

At the end of the week, on Friday, we delivered our report to the client.

# 2 Scope

Our review focused on the commit hash `f6de24d50c54471f85985e2303a04bb92c27ac71`. The list of files in scope is in the Appendix.

## 2.1 Objectives

Together with the Codefi team, we identified the following priorities for our review:

1. Ensure that the system is implemented consistently with the intended functionality and without unintended edge cases.
2. Identify known vulnerabilities particular to smart contract systems, as outlined in our Smart Contract Best Practices, and the Smart Contract Weakness Classification Registry.

3. Investigate ways to reduce the size of the contract because it is close to 24576 bytes (`0x6000`) contract size limit as described in EIP-170.
4. Make sure the contract extensions do not create problems.
5. Gas optimizations.

# 3 Security Specification

This section describes, **from a security perspective**, the expected behavior of the system under audit. It is not a substitute for documentation. The purpose of this section is to identify specific security properties that were validated by the audit team.

## 3.1 Actors

The relevant actors are listed below with their respective abilities:

- `Owner`
  - Deploys the contract.
  - Can transfer or renounce ownership.
  - Is initially a `Minter`, but minters can be managed separately.
  - Can renounce control over the contract.
  - Can stop the issuance of new tokens.
  - Can set controllers.
  - Can set partition controllers.
  - Can set default partitions.
  - Can set the validator contract address.
  - Can migrate definitively or not the current implementation to a new contract.
- `Minter`
  - Can issue (mint) new tokens in the first default partition or in any partition.
- `Controller`
  - Can set documents.
  - Is considered an `Authorized operator` if the contract is controllable.
  - Is considered an `Operator by partition` if the contract is controllable.

`Controller by partition`

  - Is considered an `Operator by partition` if the contract is controllable.
- `Operator`
  - Is appointed or removed as `Operator` by a `Token owner`.
  - Can transfer[1][2] or redeem (burn) tokens in the name of the `Token holder`.
- `Operator by partition`
  - Has the same permissions the `Operator` has, but limited to a list of partitions.
- `Token owner`
  - Can transfer tokens.
  - Can approve other addresses to spend tokens on their behalf or redeem them.
  - Can redeem (burn) tokens.

# 4 Issues

Each issue has an assigned severity:

- Minor issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
- Medium issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- Major issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Critical issues are directly exploitable security vulnerabilities that need to be fixed.

## 4.1 `ERC1400ERC20` whitelist circumvents partition restrictions Critical ✓ Fixed

| Resolution |
| --- |
| This is fixed in ConsenSys/ERC1400#13. |

## Description

ERC1400/1410 enable “partially fungible tokens” in that not all tokens are equivalent. A specific use case is placing restrictions on some tokens, such as lock-up periods.

The whitelist in `ERC1400ERC20` circumvents these restrictions. When a token holder uses the ERC20 `transfer` function, tokens are transferred from that user’s “default partitions”, which a user can choose themselves by calling `ERC1410.setDefaultPartitions`. This means they can transfer tokens from any partition, and the only restriction that’s placed on the transfer is that the *recipient* must be whitelisted.

It should be noted that the comment and error message around the whitelisting feature suggests that it is meant to be applied to both the sender and recipient:

**code/contracts/token/ERC20/ERC1400ERC20.sol:L24-L30**

```
/**
 * @dev Modifier to verify if sender and recipient are whitelisted.
 */
modifier isWhitelisted(address recipient) {
  require(_whitelisted[recipient], "A3: Transfer Blocked - Sender lockup per
  _;
}
```

## Remediation

There are many possibilities, but here are concrete suggestions for addressing this:

1. Require whitelisting both the sender and recipient, and make sure that whitelisted accounts only own (and *will* only own) unrestricted tokens.
2. Make sure that the only whitelisted recipients are those that apply partition restrictions when receiving tokens. (I.e. they implement the modified ERC777 receiving hook, examine the source partition, and reject transfers that should not occur.)
3. Instead of implementing the ERC20 interface on top of the ERC1400 token, support transferring *out* of the ERC1400 token and into a standard

ERC20 token. Partition restrictions can then be applied on the ERC1400 transfer, and once ERC20 tokens are obtained, they can be transferred without restriction.

4. Don't allow token holders to set their own default partitions. Rather, have the token specify a single, unrestricted partition that is used for all ERC20 transfers.

## 4.2 Certificate controllers do not always constrain the last argument Critical ✓ Fixed

### Resolution

The existing back end already does its own ABI encoding, which means it's not vulnerable to this issue. Documentation has been added in https://gitlab.com/ConsenSys/client/fr/dauriel/smart-contracts/certificate-controller/merge_requests/9 to ensure future maintainers understand this potential issue.

### Description

The certificate controllers ( `CertificateControllerNonce` and `CertificateControllerSalt` ) are used by passing a signature as a final argument in a function call. This signature is over the other arguments to the function. Specifically, the signature must match the call data that precedes the signature.

The way this is implemented assumes standard ABI encoding of parameters, but there's actually some room for manipulation by a malicious user. This manipulation can allow the user to change some of the call data without invalidating the signature.

The following code is from `CertificateControllerNonce` , but similar logic applies to `CertificateControllerSalt` :

**code2/contracts/CertificateControllerNonce.sol:L127-L134**

```
bytes memory payload;

assembly {
  let payloadsize := sub(calldatasize, 160)
  payload := mload(0x40) // allocate new memory
  mstore(0x40, add(payload, and(add(add(payloadsize, 0x20), 0x1f), not(0x1f)
  mstore(payload, payloadsize) // set length
  calldatacopy(add(add(payload, 0x20), 4), 4, sub(payloadsize, 4))
```

Here the signature is over all call data except the final 160 bytes. 160 bytes makes sense because the byte array is length 97, and it's preceded by a 32-byte size. This is a total of 129 bytes, and typical ABI encoded pads this to the next multiple of 32, which is 160.

If an attacker does *not* pad their arguments, they can use just 129 bytes for the signature or even 128 bytes if the `v` value happens to be 0. This means that when checking the signature, not only will the signature be excluded, but also the 31 or 32 bytes that come before the signature. This means the attacker can call a function with a different final argument than the one that was signed.

That final argument is, in many cases, the number of tokens to transfer, redeem, or issue.

## Mitigating factors

For this to be exploitable, the attacker has to be able to obtain a signature over shortened call data.

If the signer accepts raw arguments and does its own ABI encoding with standard padding, then there's likely no opportunity for an attacker to exploit this vulnerability. (They can shorten the call data length when they make the function call later, but the signature won't match.)

## Remediation

We have two suggestions for how to address this:

1. Instead of signatures being checked directly against call data, compute a new hash based on the decoded values, e.g. `keccak256(abi.encode(argument1, argument2, ...))`.

2. Address this at the signing layer (off chain) by doing the ABI encoding there and denying an attacker the opportunity to construct their own call data.

## 4.3 Salt-based certificate controller is subject to signature replay Critical ✓ Fixed

### Resolution

This is fixed in https://gitlab.com/ConsenSys/client/fr/dauriel/smart-contracts/certificate-controller/merge_requests/8.

### Description

The salt-based certificate controller prevents signature replay by storing each full signature. Only a signature that is exactly identical to a previously-used signature will be rejected.

For ECDSA signatures, each signature has a *second* S value (and flipped V to match) that will recover the same address. An attacker can produce such a second signature trivially without knowing the signer's private key. This gives an attacker a way to produce a new unique signature based on a previously used one. This effectively means every signature can be used twice.

#### code2/contracts/CertificateControllerSalt.sol:L25-L32

```
modifier isValidCertificate(bytes memory data) {

  require(
    _certificateSigners[msg.sender] || _checkCertificate(data, 0, 0x00000000
    "A3: Transfer Blocked - Sender lockup period not ended"
  );

  _usedCertificate[data] = true; // Use certificate
```

### References

- https://smartcontractsecurity.github.io/SWC-registry/docs/SWC-117.

### Remediation

Instead of rejecting used signatures based on the full signature value, keep track of used *salts* (which are then better referred to as “nonces”).

## 4.4 EIP-1400 is missing `canTransfer*` functions Major

Acknowledged

### Description

The EIP-1400 states defines the interface to be implemented containing the 3 functions:

```
// Transfer Validity
function canTransfer(address _to, uint256 _value, bytes _data) external vi
function canTransferFrom(address _from, address _to, uint256 _value, bytes
function canTransferByPartition(address _from, address _to, bytes32 _parti
```

These functions were not implemented in `ERC1400`, thus making the implementation not completely compatible with EIP-1400.

In case the deployed contract needs to be added as a “lego block” part of a another application, there is a high chance that it will not correctly function. That external application could potentially call the EIP-1400 functions `canTransfer`, `canTransferFrom` or `canTransferByPartition`, in which case the transaction will likely fail.

This means that the current implementation will not be able to become part of external markets, exchanges or applications that need to interact with a generic EIP-1400 implementation.

### Remediation

Even if the functions do not correctly reflect the transfer possibility, their omission can break other contracts interacting with the implementation.

A suggestion would be to add these functions and make them always return `true`. This way the contracts interacting with the current implementation do not break when they call these functions, while the actual transfer of the tokens is still limited by the current logic.

## 4.5 ERC777 incompatibilities Major ✓ Fixed

### Resolution

This is fixed in `ConsenSys/ERC1400#26`.

### Description

As noted in the README, the `ERC777` contract is not actually compatible with ERC 777.

Functions and events have been renamed, and the hooks `ERC777TokensRecipient` and `ERC777TokensSender` have been modified to add a `partition` parameter.

This means no tools that deal with standard ERC 777 contracts will work with this code's tokens.

### Remediation

We suggest renaming these contracts to not use the term "ERC777", as they lack compatibility. Most importantly, we recommend *not* using the interface names "ERC777TokensRecipient" and "ERC777TokensSender" when looking up the appropriate hook contracts via ERC 1820. Contracts that handle that interface will not be capable of handling the modified interface used here.

## 4.6 Buffer over-read in `ERC1410._getDestinationPartition` Major ✓ Fixed

### Resolution

This is fixed in ConsenSys/ERC1400#16.

### Description

There's no check that `data` is at least 64 bytes long, so the following code can read past the end of `data`:

**code/contracts/token/ERC1410/ERC1410.sol:L348-L361**

```
function _getDestinationPartition(bytes32 fromPartition, bytes memory data)
  bytes32 changePartitionFlag = 0xffffffffffffffffffffffffffffffffffffffffffffffff
  bytes32 flag;
  assembly {
    flag := mload(add(data, 32))
  }
  if(flag == changePartitionFlag) {
    assembly {
      toPartition := mload(add(data, 64))
    }
  } else {
    toPartition = fromPartition;
  }
}
```

The only caller is `_transferByPartition`, which only checks that `data.length > 0`:

**code/contracts/token/ERC1410/ERC1410.sol:L263-L264**

```
if(operatorData.length != 0 && data.length != 0) {
  toPartition = _getDestinationPartition(fromPartition, data);
```

Depending on how the compiler chooses to lay out memory, the next data in memory is probably the `operatorData` buffer, so data may inadvertently be read from there.

### Remediation

Check for sufficient length (at least 64 bytes) before attempting to read it.

## 4.7 ERC20/ERC777 compatibility: ERC20 transfer functions should not revert if the recipient is a contract without a registered ERC777TokensRecipient implementation Medium

✓ Fixed

### Resolution

This is fixed in ConsenSys/ERC1400#17.

### Description

The ERC20 functions `ERC1400ERC20.transfer` and `ERC1400ERC20.transferFrom` call `ERC1410._transferByDefaultPartitions`, which calls `ERC1410._transferByPartition`, which calls `ERC777._transferWithData` with the `preventLocking` argument of `true`.

This will block transfers to a contract that doesn't have an `ERC777TokensRecipient` implementation. This is in violation of ERC 777, which says:

> If the recipient is a contract, which has not registered an `ERC777TokensRecipient` implementation; then the token contract:
>
> - MUST `revert` if the `tokensReceived` hook is called from a mint or send call.
> - SHOULD continue processing the transaction if the `tokensReceived` hook is called from an ERC20 `transfer` or `transferFrom` call.

### Remediation

Make sure that ERC20-compatible transfer calls do not set `preventLocking` to `true`.

## 4.8 ERC777 compatibility: `authorizeOperator` and `revokeOperator` should revert when the caller and operator are the same account Medium ✓ Fixed

### Resolution

This is fixed in ConsenSys/ERC1400#19.

### Description

From ERC 777:

> NOTE: The *holder* (`msg.sender`) is always an `operator` for itself. This right SHALL NOT be revoked. Hence this function MUST `revert` if it is called to authorize the holder (`msg.sender`) as an *operator* for itself (i.e. if `operator` is equal to `msg.sender`).

The `autohrizeOperator` implementation does not do that:

**code/contracts/token/ERC777/ERC777.sol:L144-L147**

```
function authorizeOperator(address operator) external {
  _authorizedOperator[operator][msg.sender] = true;
  emit AuthorizedOperator(operator, msg.sender);
}
```

The same holds for `revokeOperator`:

**code/contracts/token/ERC777/ERC777.sol:L155-L158**

```
function revokeOperator(address operator) external {
  _authorizedOperator[operator][msg.sender] = false;
  emit RevokedOperator(operator, msg.sender);
}
```

### Remediation

Add `require(operator != msg.sender)` to those two functions.

## 4.9 Token receiver can mint gas tokens with sender's gas

Minor Acknowledged

### Description

When a transfer is executed, there are hooks activated on the sender's and on the receiver's side.

This is possible because the contract implements `ERC1820Client` which allows any address to define an implementation:

**contracts/ERC1820Client.sol:L16-L19**

```
function setInterfaceImplementation(string memory _interfaceLabel, address _
    bytes32 interfaceHash = keccak256(abi.encodePacked(_interfaceLabel));
    ERC1820REGISTRY.setInterfaceImplementer(address(this), interfaceHash, _i
}
```

Considering the receiver's side:

**contracts/ERC1400.sol:L1016-L1020**

```
recipientImplementation = interfaceAddr(to, ERC1400_TOKENS_RECIPIENT);

if (recipientImplementation != address(0)) {
  IERC1400TokensRecipient(recipientImplementation).tokensReceived(msg.sig, p
}
```

The sender has to pay for the gas for the transaction to go through.

Because the receiver can define a contract to be called when receiving the tokens, and the sender has to pay for the gas, the receiver can mint gas tokens (or waste the gas).

### Remediation

Because this is the way Ethereum works and the implementation allows calling external methods, there's no recommended remediation for this issue. It's just something the senders need to be aware of.

## 4.10 Missing ERC Functions Minor ✓ Fixed

| Resolution |
| --- |
| This is fixed in ConsenSys/ERC1400#18. |

### Description

There exist some functions, such as `isOperator()` ,that are part of the ERC1410 spec. Removing functions expected by ERC may break things like block

explorers that expect to be able to query standard contracts for relevant metadata.

### Remediation

It would be good to explicitly state any expected incompatibilities.

## 4.11 Inaccurate error message in `ERC777ERC20.approve`

Minor ✓ Fixed

#### Resolution

This is fixed in ConsenSys/ERC1400#20.

### Description

If the *spender* is address `0`, the revert message says that the *receiver* is not eligible.

**code/contracts/token/ERC20/ERC777ERC20.sol:L153**

```
require(spender != address(0), "A6: Transfer Blocked - Receiver not eligible
```

### Remediation

Fix the revert message to match the actual issue.

## 4.12 Non-standard treatment of a `from` address of `0` Minor

✓ Fixed

#### Resolution

This is fixed in ConsenSys/ERC1400#21.

## Description

A number of functions throughout the system treat a `from` address of `0` as equivalent to `msg.sender`. In some cases, this seems to violate existing standards (e.g. in ERC20 transfers). In other cases, it is merely surprising.

`ERC1400ERC20.transferFrom` and `ERC777ERC20.transferFrom` both treat a `from` address as `0` as equivalent to `msg.sender`. This is unexpected behavior for an ERC20 token.

## Examples

### code/contracts/ERC1400.sol:L206-L214

```
function canOperatorTransferByPartition(bytes32 partition, address from, adc
  external
  view
  returns (byte, bytes32, bytes32)
{
  if(!_checkCertificate(operatorData, 0, 0x8c0dee9c)) { // 4 first bytes of
    return(hex"A3", "", partition); // Transfer Blocked - Sender lockup perio
  } else {
    address _from = (from == address(0)) ? msg.sender : from;
```

### code/contracts/ERC1400.sol:L417-L421

```
function redeemFrom(address from, uint256 value, bytes calldata data, bytes
  external
  isValidCertificate(operatorData)
{
  address _from = (from == address(0)) ? msg.sender : from;
```

### code/contracts/token/ERC20/ERC1400ERC20.sol:L180-L181

```
function transferFrom(address from, address to, uint256 value) external isWh
  address _from = (from == address(0)) ? msg.sender : from;
```

### le/contracts/token/ERC20/ERC777ERC20.sol:L179-L180

```
function transferFrom(address from, address to, uint256 value) external isWh
  address _from = (from == address(0)) ? msg.sender : from;
```

### code/contracts/token/ERC777/ERC777.sol:L194-L198

```
function transferFromWithData(address from, address to, uint256 value, bytes
  external
  isValidCertificate(operatorData)
{
  address _from = (from == address(0)) ? msg.sender : from;
```

### code/contracts/token/ERC777/ERC777.sol:L226-L230

```
function redeemFrom(address from, uint256 value, bytes calldata data, bytes
  external
  isValidCertificate(operatorData)
{
  address _from = (from == address(0)) ? msg.sender : from;
```

### code/contracts/token/ERC1410/ERC1410.sol:L130-L142

```
function operatorTransferByPartition(
  bytes32 partition,
  address from,
  address to,
  uint256 value,
  bytes calldata data,
  bytes calldata operatorData
)
  external
  isValidCertificate(operatorData)
  returns (bytes32)
{
  address _from = (from == address(0)) ? msg.sender : from;
```

### code/contracts/token/ERC1410/ERC1410.sol:L430-L434

```
function transferFromWithData(address from, address to, uint256 value, bytes
  external
  isValidCertificate(operatorData)
{
  address _from = (from == address(0)) ? msg.sender : from;
```

### Remediation

Remove this fallback logic and always use the `from` address that was passed in. This avoids surprises where, for example, an uninitialized value leads to loss of funds.

## 4.13 `ERC1410`'s `redeem` and `redeemFrom` should revert

Minor ✓ Fixed

| Resolution |
| --- |
| This is fixed in ConsenSys/ERC1400#22. |

### Description

`ERC1410` contains two functions: `redeem` and `redeemFrom` that "erase" the underlying `ERC777` versions of these functions because those functions don't handle partitions.

These functions silently succeed, while they should probably fail by reverting.

### Examples

**code/contracts/token/ERC1410/ERC1410.sol:L441-L453**

```
/**
 * [NOT MANDATORY FOR ERC1410 STANDARD][OVERRIDES ERC777 METHOD]
 * @dev Empty function to erase ERC777 redeem() function since it doesn't hand
 */
function redeem(uint256 /*value*/, bytes calldata /*data*/) external { // Co
}

/**
 * [NOT MANDATORY FOR ERC1410 STANDARD][OVERRIDES ERC777 METHOD]
 * @dev Empty function to erase ERC777 redeemFrom() function since it doesn't
 */
function redeemFrom(address /*from*/, uint256 /*value*/, bytes calldata /*da
}
```

### Remediation

Add a `revert()` (possibly with a reason) so callers know that the call failed.

## 4.14 Unclear why `operatorData.length` is checked in `_transferByPartition` Minor ✓ Fixed

### Resolution

This code is actually correct. When `data` is present but `operatorData` is not, the `data` is the certificate (signature), which should not be used to determine the destination partition. When `operatorData` *is* present, the certificate is found there, and `data` can be used to determine the destination partition. This code checks correctly for that condition.

### Description

It's unclear why `operatorData.length` is being checked here:

**code/contracts/token/ERC1410/ERC1410.sol:L263-L264**

```
if(operatorData.length != 0 && data.length != 0) {
  toPartition = _getDestinationPartition(fromPartition, data);
```

## Remediation

Consider removing that check.

## 4.15 Global partition enumeration can run into gas limits

Minor ✓ Fixed

### Resolution

This is fixed in ConsenSys/ERC1400#25.

## Description

In `ERC1410`, partitions are created on demand by issuing or transferring tokens, and these new partitions are added to the array `_totalPartitions`. When one of these partitions is later emptied, it's removed from that array with the following code in `_removeTokenFromPartition`:

**code/contracts/token/ERC1410/ERC1410.sol:L303-L313**

```
// If the total supply is zero, finds and deletes the partition.
if(_totalSupplyByPartition[partition] == 0) {
  for (uint i = 0; i < _totalPartitions.length; i++) {
    if(_totalPartitions[i] == partition) {
      _totalPartitions[i] = _totalPartitions[_totalPartitions.length - 1];
      delete _totalPartitions[_totalPartitions.length - 1];
      _totalPartitions.length--;
      break;
    }
  }
}
```

Finding the partition requires iterating over the entire array. This means that `_removeTokenFromPartition` can become very expensive and eventually bump up against the block gas limit if lots of partitions are created. This could be an attack vector for a malicious operator.

same issue applies to a token holder's list of partitions, where transferring tokens in a large number of partitions to that token holder may

block them from being able to transfer tokens out:

**code/contracts/token/ERC1410/ERC1410.sol:L291-L301**

```
// If the balance of the TokenHolder's partition is zero, finds and deletes th
if(_balanceOfByPartition[from][partition] == 0) {
  for (uint i = 0; i < _partitionsOf[from].length; i++) {
    if(_partitionsOf[from][i] == partition) {
      _partitionsOf[from][i] = _partitionsOf[from][_partitionsOf[from].lengt
      delete _partitionsOf[from][_partitionsOf[from].length - 1];
      _partitionsOf[from].length--;
      break;
    }
  }
}
```

### Remediation

Removing an item from a set can be accomplished in constant time if the set uses both an array (for storing the values) and a mapping of values to their index in that array. See https://programtheblockchain.com/posts/2018/06/03/storage-patterns-set/ for one example of doing this.

It also may be reasonable to cap the number of possible partitions or lock them down to a constant set of values on deployment, depending on the use case for the token.

## 4.16 Optimization: redundant delete in `ERC1400._removeTokenFromPartition` Minor ✓ Fixed

| Resolution |
| --- |
| This is fixed in ConsenSys/ERC1400#23. |

### Description

Reducing the size of an array automatically deletes the removed elements, so the first of these two lines is redundant:

**code/contracts/token/ERC1410/ERC1410.sol:L296-L297**

```
delete _partitionsOf[from][_partitionsOf[from].length - 1];
_partitionsOf[from].length--;
```

The same applies here:

**code/contracts/token/ERC1410/ERC1410.sol:L308-L309**

```
delete _totalPartitions[_totalPartitions.length - 1];
_totalPartitions.length--;
```

### Remediation

Remove the redundant deletions to save a little gas.

## 4.17 Avoid hardcoding function selectors Minor ✓ Fixed

| Resolution |
| --- |
| This is fixed in ConsenSys/ERC1400#24. |

### Description

In `ERC1400`, hardcoded function selectors can be replaced with `this.transferByPartition.selector` and `this.operatorTransferByPartition.selector`.

### Examples

**code/contracts/ERC1400.sol:L184**

```
if(!_checkCertificate(data, 0, 0xf3d490db)) { // 4 first bytes of keccak256(
```

**code/contracts/ERC1400.sol:L211**

```
if(!_checkCertificate(operatorData, 0, 0x8c0dee9c)) { // 4 first bytes of ke
```

### Remediation

Replace the hardcoded function selectors with `this.<method>.selector`.

# Appendix 1 - Recommendations and Suggestions

## A.1.1 Contract Size Limit

The client said they are reaching the contract limits implemented by EIP-170 and might hit problems in the future. The current hard limit is `24576 bytes`.

The deployed bytecode of the `ERC1400` contract is `17677 bytes`. However, the contract is made to be extended by other contracts that could hit the contract size limit imposed on Ethereum. Such implementations are `ERC1400CertificateNonce` and `ERC1400CertificateSalt`, which currently hit `22464 bytes` and `22473 bytes` and are dangerously close to the hard limit.

Other implementations that extend `ERC1400` might easily go over this limit.

One such optimization, reducing the require reason strings was already implemented by the development team. Even though it reduces the readability of the error code, it reduces the contract size.

### Diamond Standard

Another, more flexible, way to reduce contract size, while also allowing for parts of the contract to be extended, is the Diamond Standard which is an extension of Transparent Contract Standard. This allows the developer to move parts of the functionality into different deployed contracts, called “facets” of the diamond, and adding these “facets” to the main contract as extensions. This extended functionality uses `delegatecall` to use external runtime bytecode while using the same main storage. A detailed explanation and implementation example is in the official EIP-2535 proposal.

This way of extending the size of code poses some risks in the actual implementation of the `delegatecall` as well as careful planning on using the storage of the contract. A feature of this standard is that parts of the contract can be separately upgraded, but this also poses the risk of upgrading some code that doesn't use the same contract storage layout as the previous code. The most significant risk is overwriting or misusing the contract storage.

### Remove Wrapper Functions

The current implementation follows a more "classical" way of Object Oriented Programming pattern, where each public function calls an internal function, which in theory could be replaced by another implementation, while the public function remains the same. This does not match well with Ethereum, where code is considered immutable, code size and gas cost matters.

The internal functions can be dropped, and the body can be added directly into the public function. This way, the code size is reduced, while also reducing the gas cost. This is because there is no `jump` from the public function code to the internal function code.

Considering `isOperator` and the internal function `_isOperator`:

```
function isOperator(address operator, address tokenHolder) external view ret
    return _isOperator(operator, tokenHolder);
}

function _isOperator(address operator, address tokenHolder) internal view re
    return (operator == tokenHolder
        || _authorizedOperator[operator][tokenHolder]
        || (_isControllable && _isController[operator])
    );
}
```

Can be rewritten to:

```
function isOperator(address operator, address tokenHolder) public view retur
    return (operator == tokenHolder
        || _authorizedOperator[operator][tokenHolder]
        || (_isControllable && _isController[operator])
    );
}
```

The body of the internal function `_isOperator` was added directly as the body of the function `isOperator` and `isOperator` was changed from `external` to `public` to be called within the contract.

A list of functions which can be rewritten:

- `transferByPartition`
- `isOperator`
- `isOperatorForPartition`
- `issueByPartition`
- `redeemByPartition`
- `setControllers`
- `setPartitionControllers`
- `setHookContract`
- `migrate`

Reducing the number of internal calls will also slightly improve the gas consumption.

## A.1.2 Gas Optimizations

There are a few gas optimizations that can be done. These will not reduce the gas significantly, so only implement them (or some of them) if you feel the gas optimization is essential to you.

- `_transferWithData`

The current code looks like this:

```
require(_balances[from] >= value, "52"); // 0x52       insufficient balance

_balances[from] = _balances[from].sub(value);
```

The safemath implementation looks like this:

```
function sub(uint256 a, uint256 b, string memory errorMessage) internal pure
    require(b <= a, errorMessage);
    uint256 c = a - b;

    return c;
}
```

Which means that the check is done twice, not to underflow. To save the most gas, the safemath call can be removed in favor of simple subtraction, reducing the number of checks and jumps.

```
require(_balances[from] >= value, "52"); // 0x52       insufficient balance

_balances[from] = _balances[from] - value;
```

- `_isMultiple`

The current code looks like this:

```
function _isMultiple(uint256 value) internal view returns(bool) {
    return(value.div(_granularity).mul(_granularity) == value);
}
```

Which is equivalent to:

```
function _isMultiple(uint256 value) internal view returns(bool) {
    return ((value % _granularity) == 0);
}
```

Which reduces the number of checks and jumps significantly.

- `operatorTransferByPartition`

current code looks like this:

```
if(_allowedByPartition[partition][from][msg.sender] >= value) {
    _allowedByPartition[partition][from][msg.sender] = _allowedByPartition[p
```

It can be rewritten without safemath, because the `if` condition already checks for underflows.

```
if(_allowedByPartition[partition][from][msg.sender] >= value) {
    _allowedByPartition[partition][from][msg.sender] = _allowedByPartition[p
```

- `transferFrom`

Similar for:

```
if(_allowed[from][msg.sender] >= value) {
    _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
```

Can be rewritten to:

```
if(_allowed[from][msg.sender] >= value) {
    _allowed[from][msg.sender] = _allowed[from][msg.sender] - value;
```

- `_removeTokenFromPartition`

The current code looks like this:

```
_balanceOfByPartition[from][partition] = _balanceOfByPartition[from][partiti
_totalSupplyByPartition[partition] = _totalSupplyByPartition[partition].sub(
```

There are 2 instances where `_removeTokenFromPartition` is called:

In `_redeemByPartition`:

```
require(_balanceOfByPartition[from][fromPartition] >= value, "52"); // 0x52

[...]

_removeTokenFromPartition(from, fromPartition, value);
```

And in `_transferByPartition`:

```
require(_balanceOfByPartition[from][fromPartition] >= value, "52"); // 0x52

[...]

_removeTokenFromPartition(from, fromPartition, value);
```

And in both cases the `value` has to be smaller or equal to the balance, in which case no underflow can happen.

However, because there is a disconnect between where the check is done and where the subtraction happens, removing the safemath, even though it reduces the gas cost, makes the developer remember the `_removeTokenFromPartition` does not do a safe subtraction. This can be a problem if the developer needs to add more functionality and should remember the internal function does not do the necessary checks.

**Observation**

Other gas optimization techniques can be implemented. However, they might introduce significant rewrites or increase the code size. Hence they were not recommended in this case.

Also, with these suggestions, you should implement them only if you are sure you understand all the implications of reducing the gas cost, removing additional checks and gas reduced is significant for your application.

## A.1.3 Error Codes

The EIP-1400 states that the function `canTransfer` MUST return a reason byte code on success or failure based on EIP-1066 error codes. This creates an

inconsistency between these error codes and the other errors returned in the contract.

In order to have more consistency and a smaller contract size, the development team changed all the error strings to error codes in the rest of the implementation.

### Examples

Error codes as strings in `require`s:

```
modifier isIssuableToken() {
    require(_isIssuable, "55"); // 0x55        funds locked (lockup period)
    _;
}
```

```
function transferFrom(address from, address to, uint256 value) external retu
    require( _isOperator(msg.sender, from)
      || (value <= _allowed[from][msg.sender]), "53"); // 0x53  insufficient
```

Error codes as hex argument in `_canTransfer`:

```
function _canTransfer(bytes4 functionSig, bytes32 partition, address operato
    internal
    view
    returns (byte, bytes32, bytes32)
{
    address checksImplementation = interfaceAddr(address(this), ERC1400_TOKE

    if((checksImplementation != address(0))) {
        return IERC1400TokensChecker(checksImplementation).canTransferByPart
    }
    else {
        return(hex"00", "", partition);
    }
}
```

## Description

Excerpt from the EIP-1066 standard:

> This provides a shared set of signals to allow smart contracts to react to situations autonomously, expose localized error messages to users, and so on.

The initial intention of returning these error codes is to allow other contracts to interpret the status instead of failing without knowing the reason why.

Because the EIP-1400 standard does not fully implement the error codes, it creates a difficult choice for the developer to have inconsistency between different parts of the code that return error strings in contrast with the `_canTransfer` method that returns hex errors.

To illustrate the difference of failing with error strings versus returning a status code we have created these test contracts.

```
contract ReturnTest {
    function returnRequireErrorString() public {
        require(false, "50");
    }

    function returnRequireErrorHex() public {
        require(false, hex"50");
    }

    function returnFirstByteArgument() public returns (byte status) {
        return hex"50";
    }
}

contract ReadReturn {
    ReturnTest public returnTest;

    event Success(bool);
    event Message(bytes);

    constructor () public {
        returnTest = new ReturnTest();
    }

    function callMethod(string calldata _method) external returns (bool, byt
        (bool success, bytes memory message) = address(returnTest).call(abi.

        emit Success(success);
        emit Message(message);
    }
}
```

Calling `ReadReturn.callMethod()` emits these events for each `_method` argument:

- `callMethod('returnRequireErrorString()')`

```
Success(false)
Message('0x08c379a0000000000000000000000000000000000000000000000000000000
```

generated by

```
require(false, "50");
```

From the returned message, `0x3530` represents the string “50” and the preceding `0x02` is the length of the returned string.

- `callMethod('returnRequireErrorHex()')`

```
Success(false)
Message('0x08c379a0000000000000000000000000000000000000000000000000000000
```

generated by

```
require(false, hex"50");
```

This time the returned message contains `0x50` which represents the `hex"50"` and the preceding `0x01` is the length of the returned string.

- `callMethod('returnFirstByteArgument()')`

```
Success(true)
Message('0x5000000000000000000000000000000000000000000000000000000000000
```

generated by

```
return hex"50";
```

This time the returned value contains `0x50` which is the first byte of the returned bytes, and can be easily used in another smart contract as specified by ERC-1066.

The recurring hex in the first 2 calls `0x08c379a0` is the first 4 bytes of `keccak256("Error(string)")` which is always returned when a `revert` or `require` is triggered.

### Remediation

There is no recommendation in this case because of the already described limitations of contract size and adhering to the EIP-1400 standard, an elegant solution was not identified.

We wanted to make sure we explain the difference between all approaches, specifically the difference between:

- `require(false, "50");`
- `require(false, hex"50");`
- `return hex"50";`

# Appendix 2 - Files in Scope

The commit hash that was reviewed is `f6de24d50c54471f85985e2303a04bb92c27ac71`.

This code review covered the following files:

| File Name | SHA-1 Hash |
| --- | --- |
| ./contracts/ERC1400.sol | `adc01ee17e9379e623639cd3bef68de5e5aee264` |
| ./contracts/certificate/ERC1400CertificateNonce.sol | `55baaf3be216704b6170251829708f626ea1a347` |
| ./contracts/certificate/ERC1400CertificateSalt.sol | `35e093fcbc90a086e6ed2f0b2611e4909c6f4562` |

# Appendix 3 - Disclosure

ConsenSys Diligence (“CD”) typically receives compensation from one or more clients (the “Clients”) for performing the analysis contained in these reports (the “Reports”). The Reports may be distributed through other means, including via ConsenSys publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any Third-Party by virtue of publishing these Reports.

PURPOSE OF REPORTS The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of Solidity code and only the Solidity code we note as being within the scope of our review within this report. The Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond Solidity that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty.

CD makes the Reports available to parties other than the Clients (i.e., “third parties”) – on its website. CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

LINKS TO OTHER WEB SITES FROM THIS WEB SITE You may, through hypertext or other computer links, gain access to web sites operated by persons other than ConsenSys and CD. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that ConsenSys and CD are not responsible for the content or operation of such Web sites, and that ConsenSys and CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or mean that ConsenSys and CD endorses the content on that Web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the Reports. ConsenSys and CD assumes no responsibility for the use of third party software on the Web Site and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

TIMELINESS OF CONTENT The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice. Unless indicated otherwise, by ConsenSys and CD.

# Appendix 4 - Artifacts

This section contains some of the artifacts generated during our review by automated tools, the test suite, etc. If any issues or recommendations were identified by the output presented here, they have been addressed in the appropriate section above.

## A.4.1 MythX

MythX is a security analysis API for Ethereum smart contracts. It performs multiple types of analysis, including fuzzing and symbolic execution, to detect many common vulnerability types. The tool was used for automated vulnerability discovery for all audited contracts and libraries. More details on MythX can be found at mythx.io.

Below is the raw output of the MythX vulnerability scan:

```
mythos analyze ./erc1400.sol ERC1400 --solcVersion=0.5.10 --analysisMode=f
  Reading contract ./erc1400.sol... done
```

```
Compiling with Solidity version: v0.5.10+commit.5a6ea5b1
Compiling contract ./erc1400.sol... done
Analyzing contract ERC1400... done

UUID: cbdf1e9f-6ca2-4ddd-ba15-4491aa844ab8
API Version: v1.8.0.1
Harvey Version: 0.0.43
Maestro Version: undefined
Maru Version: 0.7.8
Mythril Version: 0.22.2

Report found: 14 issues
Covered instructions: undefined
Covered paths: undefined
Selected compiler version: vUnknown

Title: (SWC-000) Unknown
Severity: Medium
Head: Incorrect function "_canTransfer" state mutability
Description: Function "_canTransfer" state mutability is considered "view" b
Source code:

./erc1400.sol 1834:4
-------------------------------------------------
nction _canTransfer(bytes4 functionSig, bytes32 partition, address operator,
    internal
    view
    returns (byte, bytes32, bytes32)
  {
    address checksImplementation = interfaceAddr(address(this), ERC1400_TOKE

    if((checksImplementation != address(0))) {
      return IERC1400TokensChecker(checksImplementation).canTransferByPartit
    }
    else {
      return(hex"00", "", partition);
    }
  }

-------------------------------------------------

=================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
ource code:

./erc1400.sol 4:0
```

```
--------------------------------------------------
pragma solidity ^0.5.0;
--------------------------------------------------

==================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./erc1400.sol 72:0
--------------------------------------------------
pragma solidity ^0.5.0;
--------------------------------------------------

==================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./erc1400.sol 147:0
--------------------------------------------------
pragma solidity ^0.5.0;
--------------------------------------------------

==================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./erc1400.sol 173:0
--------------------------------------------------
pragma solidity ^0.5.0;
--------------------------------------------------

==================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:
```

```
Source code:

./erc1400.sol 216:0
------------------------------------------------
pragma solidity ^0.5.0;
------------------------------------------------

================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.3"". It is recc
Source code:

./erc1400.sol 261:0
------------------------------------------------
pragma solidity ^0.5.3;
------------------------------------------------

================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./erc1400.sol 297:0
------------------------------------------------
pragma solidity ^0.5.0;
------------------------------------------------

================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./erc1400.sol 329:0
------------------------------------------------
pragma solidity ^0.5.0;
------------------------------------------------

================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
```

```
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./erc1400.sol 478:0
-----------------------------------------------
pragma solidity ^0.5.0;
-----------------------------------------------

===============================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./erc1400.sol 516:0
-----------------------------------------------
pragma solidity ^0.5.0;
-----------------------------------------------

===============================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./erc1400.sol 553:0
-----------------------------------------------
pragma solidity ^0.5.0;
-----------------------------------------------

===============================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./erc1400.sol 591:0
-----------------------------------------------
pragma solidity ^0.5.0;
-----------------------------------------------

===============================================
```

```
Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./erc1400.sol 629:0
-----------------------------------------------
pragma solidity ^0.5.0;
-----------------------------------------------

===============================================

Done
```

```
$ mythos analyze ./ERC1400CertificateNonce.sol ERC1400CertificateNonce --sol
Reading contract ./ERC1400CertificateNonce.sol... done
Compiling with Solidity version: v0.5.10+commit.5a6ea5b1
Compiling contract ./ERC1400CertificateNonce.sol... done
Analyzing contract ERC1400CertificateNonce... done

UUID: 69771148-5afb-46ac-aab5-1b3493219819
API Version: v1.8.0.1
Harvey Version: 0.0.43
Maestro Version: undefined
Maru Version: 0.7.8
Mythril Version: 0.22.2

Report found: 17 issues
Covered instructions: undefined
Covered paths: undefined
Selected compiler version: vUnknown

Title: (SWC-000) Unknown
Severity: Medium
Head: Incorrect function "_canTransfer" state mutability
Description: Function "_canTransfer" state mutability is considered "view" b
Source code:

./ERC1400CertificateNonce.sol 1834:4
-----------------------------------------------
nction _canTransfer(bytes4 functionSig, bytes32 partition, address operator,
    internal
    view
    returns (byte, bytes32, bytes32)
{
    address checksImplementation = interfaceAddr(address(this), ERC1400_TOKE
```

```
    if((checksImplementation != address(0))) {
      return IERC1400TokensChecker(checksImplementation).canTransferByPartit
    }
    else {
      return(hex"00", "", partition);
    }
  }

-----------------------------------------------

===============================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateNonce.sol 4:0
-----------------------------------------------
pragma solidity ^0.5.0;
-----------------------------------------------

===============================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateNonce.sol 72:0
-----------------------------------------------
pragma solidity ^0.5.0;
-----------------------------------------------

===============================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateNonce.sol 147:0
-----------------------------------------------
pragma solidity ^0.5.0;
-----------------------------------------------
```

```
==================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateNonce.sol 173:0
--------------------------------------------------
pragma solidity ^0.5.0;
--------------------------------------------------

==================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateNonce.sol 216:0
--------------------------------------------------
pragma solidity ^0.5.0;
--------------------------------------------------

==================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.3"". It is recc
Source code:

./ERC1400CertificateNonce.sol 261:0
--------------------------------------------------
pragma solidity ^0.5.3;
--------------------------------------------------

==================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateNonce.sol 297:0
--------------------------------------------------
```

```
pragma solidity ^0.5.0;
-----------------------------------------------

===============================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateNonce.sol 329:0
-----------------------------------------------
pragma solidity ^0.5.0;
-----------------------------------------------

===============================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateNonce.sol 478:0
-----------------------------------------------
pragma solidity ^0.5.0;
-----------------------------------------------

===============================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateNonce.sol 516:0
-----------------------------------------------
pragma solidity ^0.5.0;
-----------------------------------------------

===============================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:
```

```
./ERC1400CertificateNonce.sol 553:0
------------------------------------------------
pragma solidity ^0.5.0;
------------------------------------------------

================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateNonce.sol 591:0
------------------------------------------------
pragma solidity ^0.5.0;
------------------------------------------------

================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateNonce.sol 629:0
------------------------------------------------
pragma solidity ^0.5.0;
------------------------------------------------

================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateNonce.sol 1932:2
------------------------------------------------
agma solidity ^0.5.0;


------------------------------------------------

================================================

itle: (SWC-103) Floating Pragma
Severity: Low
```

```
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateNonce.sol 2120:2
-----------------------------------------------
agma solidity ^0.5.0;


-----------------------------------------------

===============================================

Title: (SWC-108) State Variable Default Visibility
Severity: Low
Head: State variable visibility is not set.
Description: It is best practice to set the visibility of state variables ex
Source code:

./ERC1400CertificateNonce.sol 1939:7
-----------------------------------------------
_certificateControllerActivated
-----------------------------------------------

===============================================

Done
```

```
$ mythos analyze ./ERC1400CertificateSalt.sol ERC1400CertificateSalt --solcV
Reading contract ./ERC1400CertificateSalt.sol... done
Compiling with Solidity version: v0.5.10+commit.5a6ea5b1
Compiling contract ./ERC1400CertificateSalt.sol... done
Analyzing contract ERC1400CertificateSalt... done

UUID: 443df106-799a-4a8a-be3e-c538ec9395e1
API Version: v1.8.0.1
Harvey Version: 0.0.43
Maestro Version: undefined
Maru Version: 0.7.8
Mythril Version: 0.22.2

Report found: 17 issues
Covered instructions: undefined
Covered paths: undefined
Selected compiler version: vUnknown
```

```
Title: (SWC-000) Unknown
Severity: Medium
Head: Incorrect function "_canTransfer" state mutability
Description: Function "_canTransfer" state mutability is considered "view" b
Source code:

./ERC1400CertificateSalt.sol 1834:4
-------------------------------------------------
nction _canTransfer(bytes4 functionSig, bytes32 partition, address operator,
    internal
    view
    returns (byte, bytes32, bytes32)
  {
    address checksImplementation = interfaceAddr(address(this), ERC1400_TOKE

    if((checksImplementation != address(0))) {
      return IERC1400TokensChecker(checksImplementation).canTransferByPartit
    }
    else {
      return(hex"00", "", partition);
    }
  }

-------------------------------------------------

=================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateSalt.sol 4:0
-------------------------------------------------
pragma solidity ^0.5.0;
-------------------------------------------------

=================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateSalt.sol 72:0
-------------------------------------------------
ragma solidity ^0.5.0;
-------------------------------------------------
```

```
==================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateSalt.sol 147:0
--------------------------------------------------
pragma solidity ^0.5.0;
--------------------------------------------------

==================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateSalt.sol 173:0
--------------------------------------------------
pragma solidity ^0.5.0;
--------------------------------------------------

==================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateSalt.sol 216:0
--------------------------------------------------
pragma solidity ^0.5.0;
--------------------------------------------------

==================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.3"". It is recc
Source code:

./ERC1400CertificateSalt.sol 261:0
--------------------------------------------------
```

```
pragma solidity ^0.5.3;
------------------------------------------------

================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateSalt.sol 297:0
------------------------------------------------
pragma solidity ^0.5.0;
------------------------------------------------

================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateSalt.sol 329:0
------------------------------------------------
pragma solidity ^0.5.0;
------------------------------------------------

================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateSalt.sol 478:0
------------------------------------------------
pragma solidity ^0.5.0;
------------------------------------------------

================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:
```

```
./ERC1400CertificateSalt.sol 516:0
------------------------------------------------
pragma solidity ^0.5.0;
------------------------------------------------

================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateSalt.sol 553:0
------------------------------------------------
pragma solidity ^0.5.0;
------------------------------------------------

================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateSalt.sol 591:0
------------------------------------------------
pragma solidity ^0.5.0;
------------------------------------------------

================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateSalt.sol 629:0
------------------------------------------------
pragma solidity ^0.5.0;
------------------------------------------------

================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
```

```
Description: The current pragma Solidity directive is ""^0.5.0"". It is rec
Source code:

./ERC1400CertificateSalt.sol 1932:2
-----------------------------------------------------
agma solidity ^0.5.0;


-----------------------------------------------------

=====================================================

Title: (SWC-103) Floating Pragma
Severity: Low
Head: A floating pragma is set.
Description: The current pragma Solidity directive is ""^0.5.0"". It is recc
Source code:

./ERC1400CertificateSalt.sol 2123:2
-----------------------------------------------------
agma solidity ^0.5.0;


-----------------------------------------------------

=====================================================

Title: (SWC-108) State Variable Default Visibility
Severity: Low
Head: State variable visibility is not set.
Description: It is best practice to set the visibility of state variables ex
Source code:

./ERC1400CertificateSalt.sol 1939:7
-----------------------------------------------------
_certificateControllerActivated
-----------------------------------------------------

=====================================================

Done
```

The scan did not reveal any security issues.

## A.4.2 Surya

Surya is a utility tool for smart contract systems. It provides a number of visual outputs and information about the structure of smart contracts. It also supports querying the function call graph in multiple ways to aid in the manual inspection and control flow analysis of contracts.

Below is a complete list of functions with their visibility and modifiers:

## Files Description Table

| File Name | SHA-1 Hash |
|---|---|
| ./contracts/ERC1400.sol | adc01ee17e9379e623639cd3bef68de5e5aee264 |
| ./contracts/certificate/ERC1400CertificateNonce.sol | 55baaf3be216704b6170251829708f626ea1a347 |
| ./contracts/certificate/ERC1400CertificateSalt.sol | 35e093fcbc90a086e6ed2f0b2611e4909c6f4562 |

## Contracts Description Table

| Contract | Type | Bases | | |
|---|---|---|---|---|
| L | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| | | | | |
| **ERC1400** | Implementation | IERC20, IERC1400, Ownable, ERC1820Client, ERC1820Implementer, MinterRole | | |
| L | <Constructor> | Public ❗ | 🛑 | |
| L | totalSupply | External ❗ | | NO❗ |
| L | balanceOf | External ❗ | | NO❗ |
| L | transfer | External ❗ | 🛑 | NO❗ |

| Contract | Type | Bases | | |
|---|---|---|---|---|
| L | allowance | External ❗ | | NO❗ |
| L | approve | External ❗ | 🛑 | NO❗ |
| L | transferFrom | External ❗ | 🛑 | NO❗ |
| L | getDocument | External ❗ | | NO❗ |
| L | setDocument | External ❗ | 🛑 | NO❗ |
| L | balanceOfByPartition | External ❗ | | NO❗ |
| L | partitionsOf | External ❗ | | NO❗ |
| L | transferWithData | External ❗ | 🛑 | NO❗ |
| L | transferFromWithData | External ❗ | 🛑 | NO❗ |
| L | transferByPartition | External ❗ | 🛑 | NO❗ |
| L | operatorTransferByPartition | External ❗ | 🛑 | NO❗ |
| L | isControllable | External ❗ | | NO❗ |
| L | authorizeOperator | External ❗ | 🛑 | NO❗ |
| L | revokeOperator | External ❗ | 🛑 | NO❗ |

| Contract | Type | Bases | | |
|---|---|---|---|---|
| └ | authorizeOperatorByPartition | External ❗ | 🛑 | NO❗ |
| └ | revokeOperatorByPartition | External ❗ | 🛑 | NO❗ |
| └ | isOperator | External ❗ | | NO❗ |
| └ | isOperatorForPartition | External ❗ | | NO❗ |
| └ | isIssuable | External ❗ | | NO❗ |
| └ | issue | External ❗ | 🛑 | onlyMinter isIssuableToken |
| └ | issueByPartition | External ❗ | 🛑 | onlyMinter isIssuableToken |
| └ | redeem | External ❗ | 🛑 | NO❗ |
| └ | redeemFrom | External ❗ | 🛑 | NO❗ |
| └ | redeemByPartition | External ❗ | 🛑 | NO❗ |
| └ | operatorRedeemByPartition | External ❗ | 🛑 | NO❗ |
| └ | name | External ❗ | | NO❗ |
| └ | symbol | External ❗ | | NO❗ |
| └ | decimals | External ❗ | | NO❗ |
| └ | granularity | External ❗ | | NO❗ |

| Contract | Type | Bases | | |
|---|---|---|---|---|
| L | totalPartitions | External ❗ | | NO❗ |
| L | renounceControl | External ❗ | 🛑 | onlyOwner |
| L | renounceIssuance | External ❗ | 🛑 | onlyOwner |
| L | controllers | External ❗ | | NO❗ |
| L | controllersByPartition | External ❗ | | NO❗ |
| L | setControllers | External ❗ | 🛑 | onlyOwner |
| L | setPartitionControllers | External ❗ | 🛑 | onlyOwner |
| L | getDefaultPartitions | External ❗ | | NO❗ |
| L | setDefaultPartitions | External ❗ | 🛑 | onlyOwner |
| L | allowanceByPartition | External ❗ | | NO❗ |
| L | approveByPartition | External ❗ | 🛑 | NO❗ |
| L | setHookContract | External ❗ | 🛑 | onlyOwner |
| L | migrate | External ❗ | 🛑 | onlyOwner |
| L | _transferWithData | Internal 🔒 | 🛑 | isNotMigratedToken |
| L | _transferByPartition | Internal 🔒 | 🛑 | |

| Contract | Type | Bases | | |
|---|---|---|---|---|
| L | _transferByDefaultPartitions | Internal 🔒 | 🛑 | |
| L | _getDestinationPartition | Internal 🔒 | | |
| L | _removeTokenFromPartition | Internal 🔒 | 🛑 | |
| L | _addTokenToPartition | Internal 🔒 | 🛑 | |
| L | _isMultiple | Internal 🔒 | | |
| L | _callPreTransferHooks | Internal 🔒 | 🛑 | |
| L | _callPostTransferHooks | Internal 🔒 | 🛑 | |
| L | _isOperator | Internal 🔒 | | |
| L | _isOperatorForPartition | Internal 🔒 | | |
| L | _issue | Internal 🔒 | 🛑 | isNotMigratedToken |
| L | _issueByPartition | Internal 🔒 | 🛑 | |
| L | _redeem | Internal 🔒 | 🛑 | isNotMigratedToken |

| Contract | Type | Bases | | |
|---|---|---|---|---|
| L | _redeemByPartition | Internal 🔒 | 🛑 | |
| L | _redeemByDefaultPartitions | Internal 🔒 | 🛑 | |
| L | _canTransfer | Internal 🔒 | | |
| L | _setControllers | Internal 🔒 | 🛑 | |
| L | _setPartitionControllers | Internal 🔒 | 🛑 | |
| L | _setHookContract | Internal 🔒 | 🛑 | |
| L | _migrate | Internal 🔒 | 🛑 | |
| | | | | |
| **ERC1400 Certificate Nonce** | Implementation | ERC1400, CertificateController | | |
| L | <Constructor> | Public ❗ | 🛑 | ERC1400 Certificate Controller |
| L | setCertificateSigner | External ❗ | 🛑 | onlyOwner |
| L | setCertificateControllerActivated | External ❗ | 🛑 | onlyOwner |
| L | transferWithData | External ❗ | 🛑 | isValidCertificate |

| Contract | Type | Bases | | |
|---|---|---|---|---|
| L | transferFromWithData | External ❗ | 🛑 | isValidCertificate |
| L | transferByPartition | External ❗ | 🛑 | isValidCertificate |
| L | operatorTransferByPartition | External ❗ | 🛑 | isValidCertificate |
| L | issue | External ❗ | 🛑 | onlyMinter isIssuableToken isValidCertificate |
| L | issueByPartition | External ❗ | 🛑 | onlyMinter isIssuableToken isValidCertificate |
| L | redeem | External ❗ | 🛑 | isValidCertificate |
| L | redeemFrom | External ❗ | 🛑 | isValidCertificate |
| L | redeemByPartition | External ❗ | 🛑 | isValidCertificate |
| L | operatorRedeemByPartition | External ❗ | 🛑 | isValidCertificate |
| L | canTransferByPartition | External ❗ | | NO❗ |

| Contract | Type | Bases | | |
|---|---|---|---|---|
| L | canOperatorTransferByPartition | External ❗ | | NO❗ |
| | | | | |
| **ERC1400 Certificate Salt** | Implementation | ERC1400, CertificateController | | |
| L | <Constructor> | Public ❗ | 🛑 | ERC1400 Certificate Controller |
| L | setCertificateSigner | External ❗ | 🛑 | onlyOwner |
| L | setCertificateControllerActivated | External ❗ | 🛑 | onlyOwner |
| L | transferWithData | External ❗ | 🛑 | isValidCertificate |
| L | transferFromWithData | External ❗ | 🛑 | isValidCertificate |
| L | transferByPartition | External ❗ | 🛑 | isValidCertificate |
| L | operatorTransferByPartition | External ❗ | 🛑 | isValidCertificate |
| L | issue | External ❗ | 🛑 | onlyMinter isIssuableToken isValidCertificate |

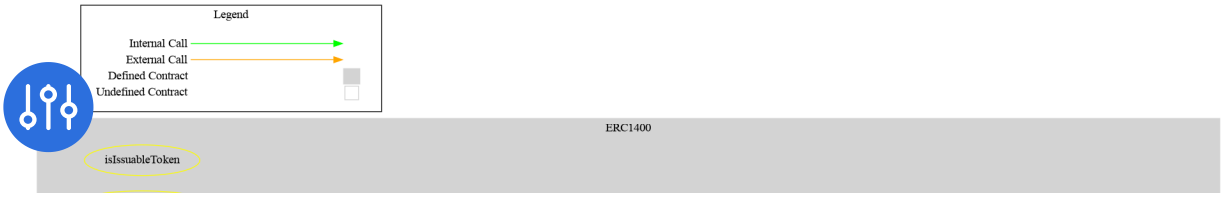
| Contract | Type | Bases | | |
|---|---|---|---|---|
| L | issueByPartition | External ❗ | 🛑 | onlyMinter isIssuableToken isValidCertificate |
| L | redeem | External ❗ | 🛑 | isValidCertificate |
| L | redeemFrom | External ❗ | 🛑 | isValidCertificate |
| L | redeemByPartition | External ❗ | 🛑 | isValidCertificate |
| L | operatorRedeemByPartition | External ❗ | 🛑 | isValidCertificate |
| L | canTransferByPartition | External ❗ | | NO❗ |
| L | canOperatorTransferByPartition | External ❗ | | NO❗ |

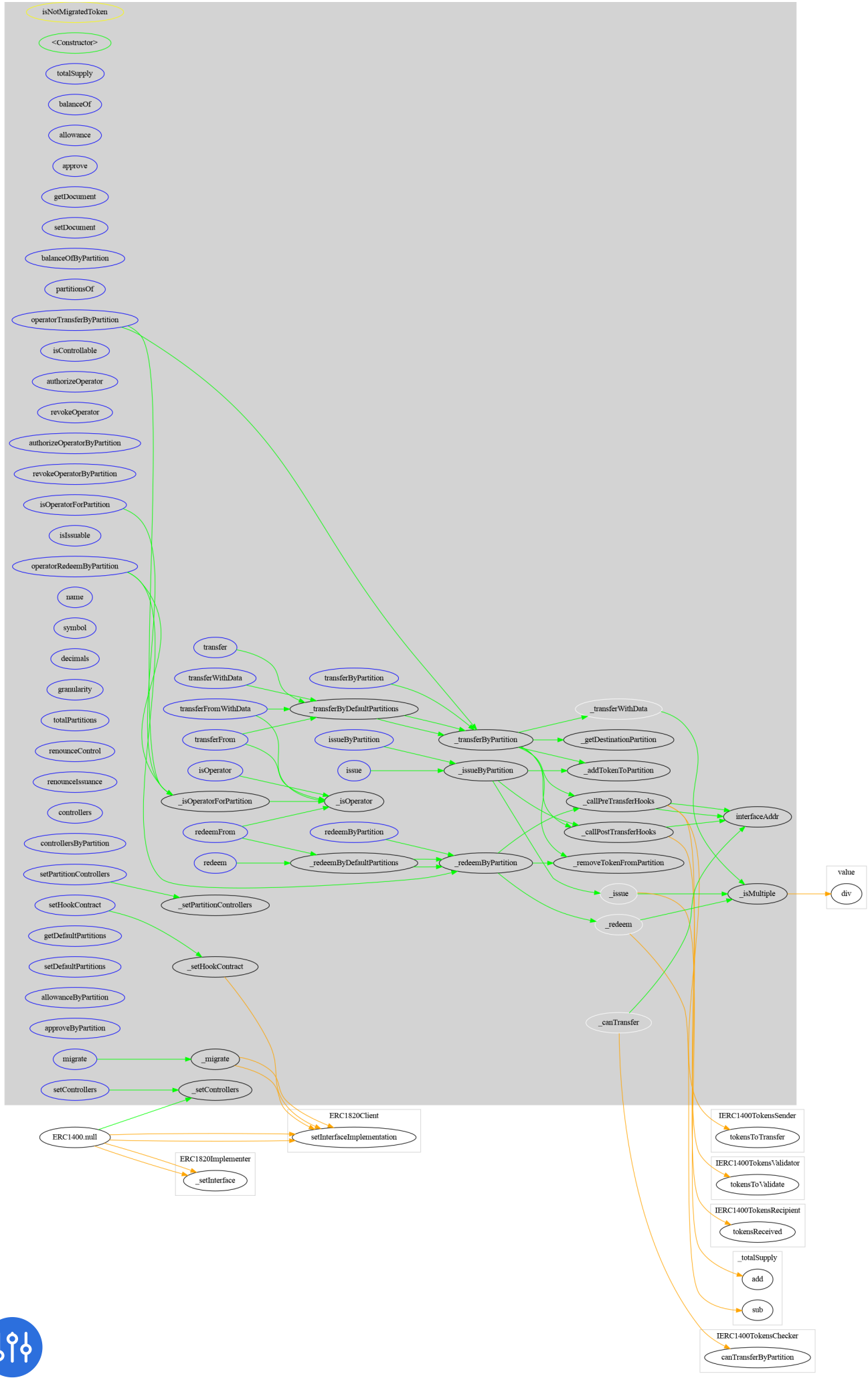
## Legend

| Symbol | Meaning |
|---|---|
| 🛑 | Function can modify state |
| 💵 | Function is payable |

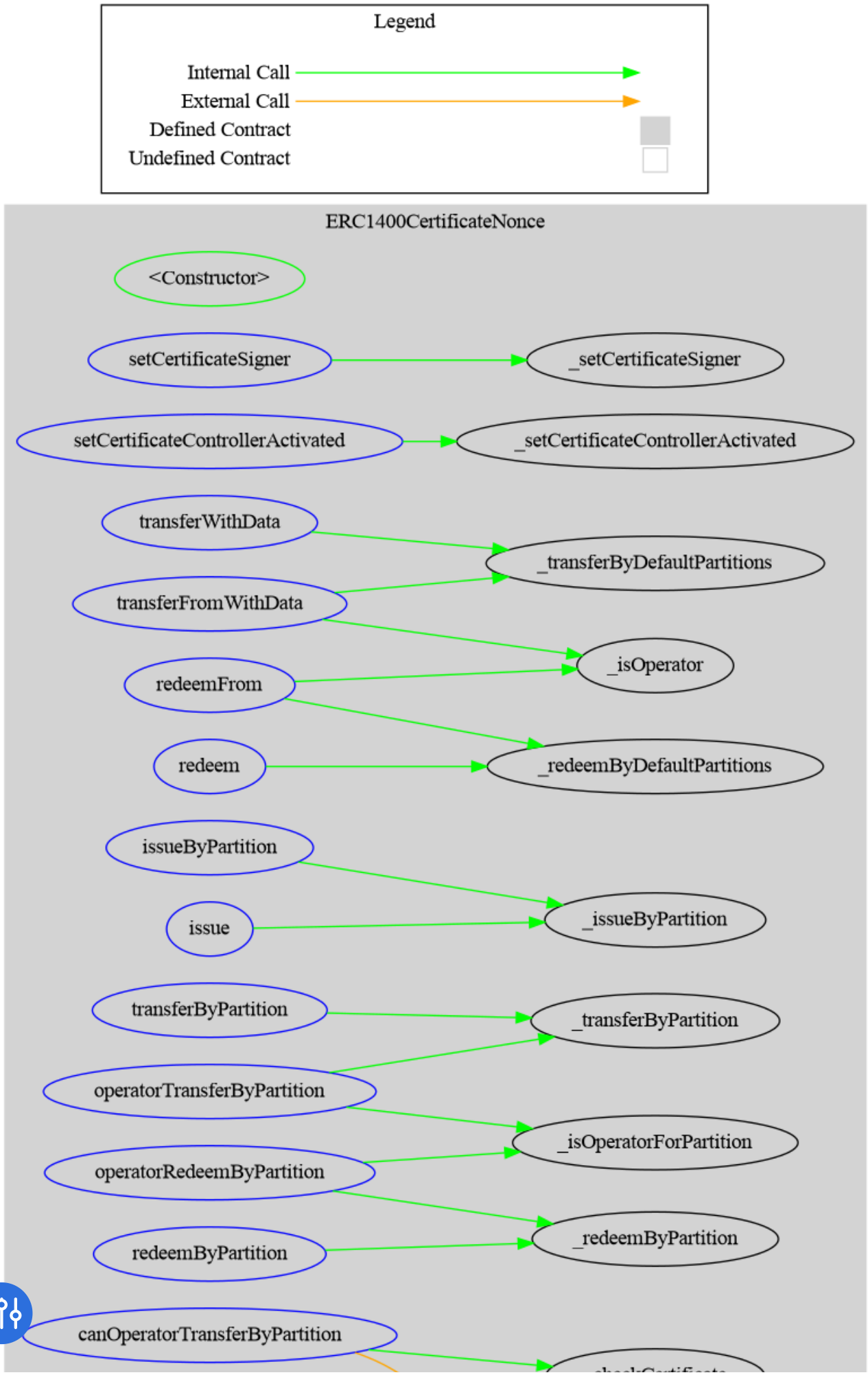
## Graph

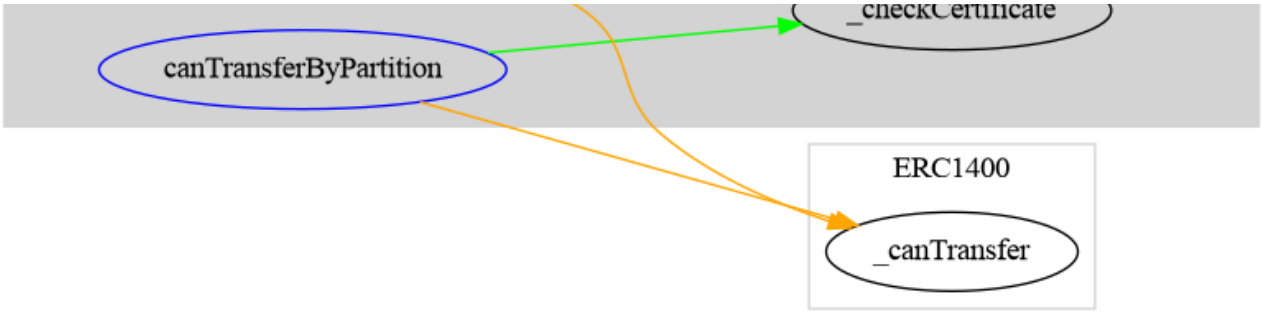
### ERC1400



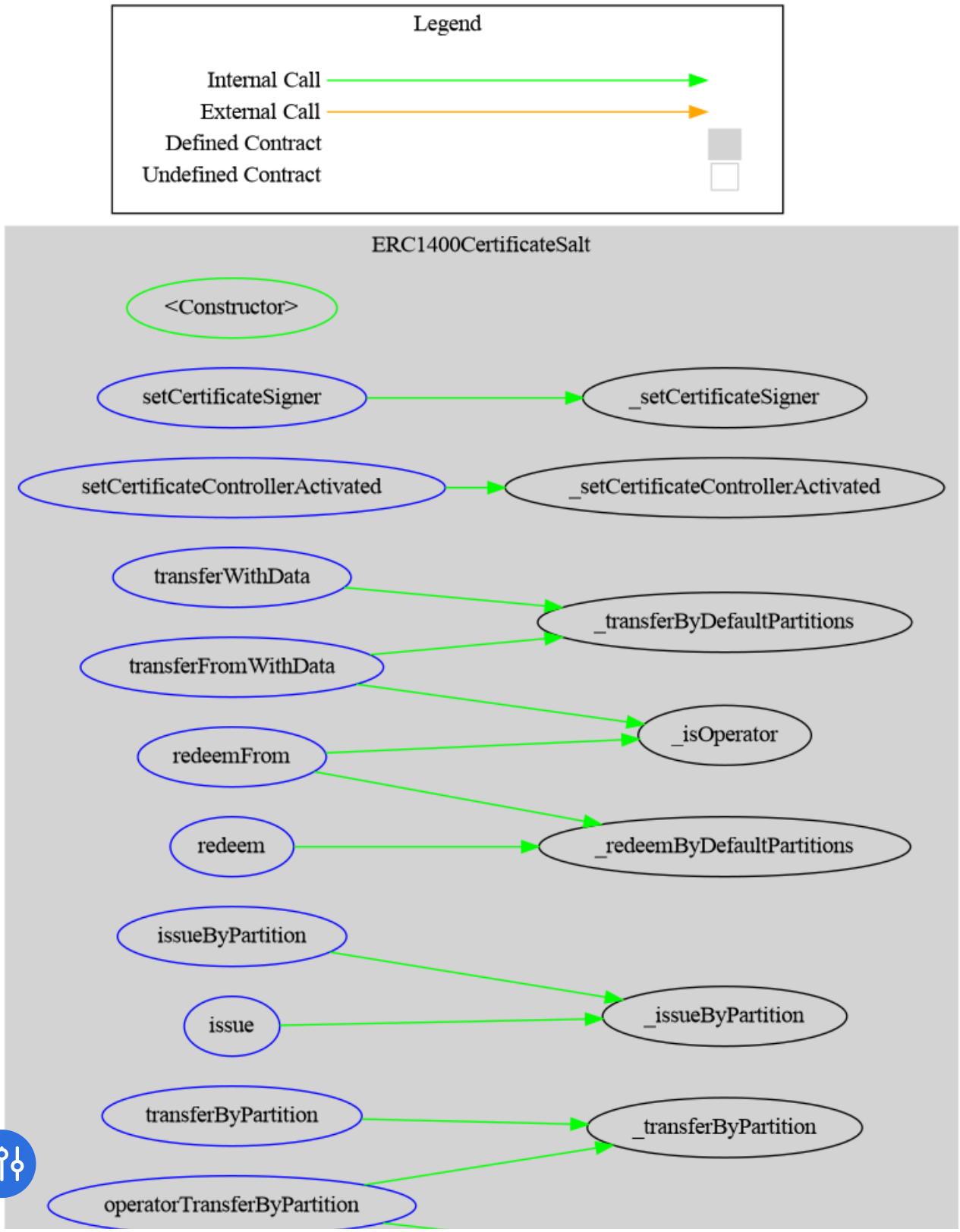

isNotMigratedToken
<Constructor>
totalSupply
balanceOf
allowance
approve
getDocument
setDocument
balanceOfByPartition
partitionsOf
operatorTransferByPartition
isControllable
authorizeOperator
revokeOperator
authorizeOperatorByPartition
revokeOperatorByPartition
isOperatorForPartition
isIssuable
operatorRedeemByPartition
name
symbol
decimals
granularity
totalPartitions
renounceControl
renounceIssuance
controllers
controllersByPartition
setPartitionControllers
setHookContract
getDefaultPartitions
setDefaultPartitions
allowanceByPartition
approveByPartition
migrate
setControllers
transfer
transferWithData
transferFromWithData
transferFrom
isOperator
_isOperatorForPartition
_isOperator
redeemFrom
redeem
transferByPartition
_transferByDefaultPartitions
issueByPartition
issue
redeemByPartition
_redeemByDefaultPartitions
_setPartitionControllers
_setHookContract
_migrate
_setControllers
_transferByPartition
_issueByPartition
_redeemByPartition
_transferWithData
_getDestinationPartition
_addTokenToPartition
_callPreTransferHooks
_callPostTransferHooks
_removeTokenFromPartition
_issue
_redeem
_canTransfer
interfaceAddr
_isMultiple
value
div
ERC1400.null
ERC1820Client
setInterfaceImplementation
ERC1820Implementer
_setInterface
IERC1400TokensSender
tokensToTransfer
IERC1400TokensValidator
tokensToValidate
IERC1400TokensRecipient
tokensReceived
_totalSupply
add
sub
IERC1400TokensChecker
canTransferByPartition

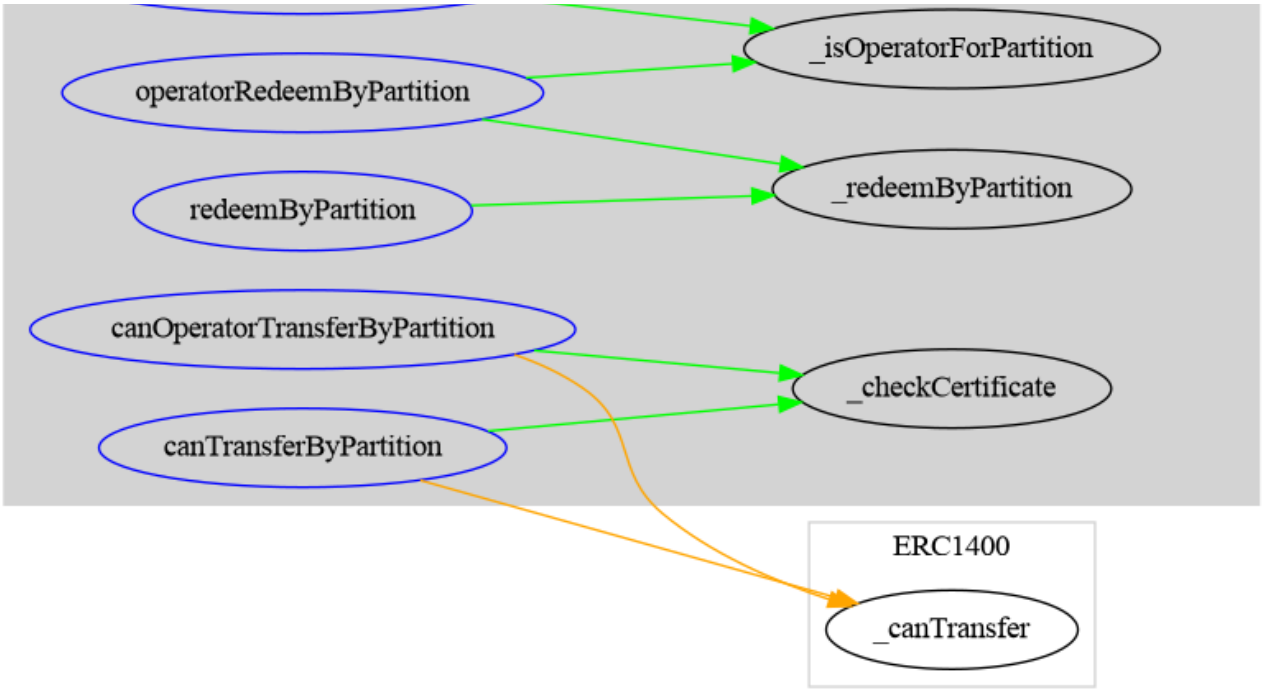
## ERC1400CertificateNonce





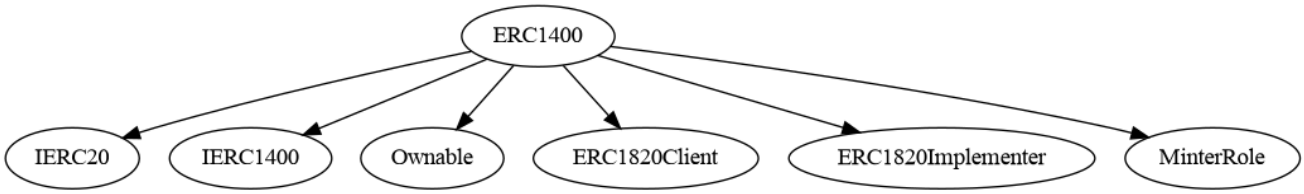

## ERC1400CertificateSalt



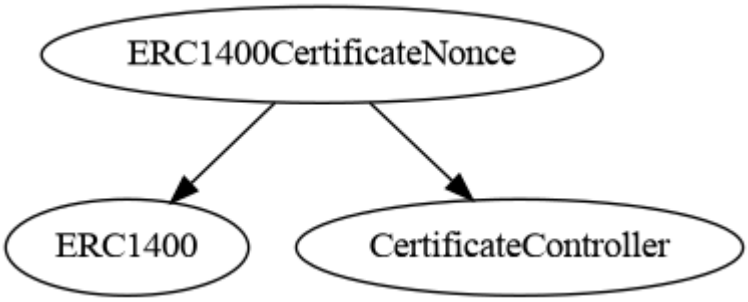



# Inheritance

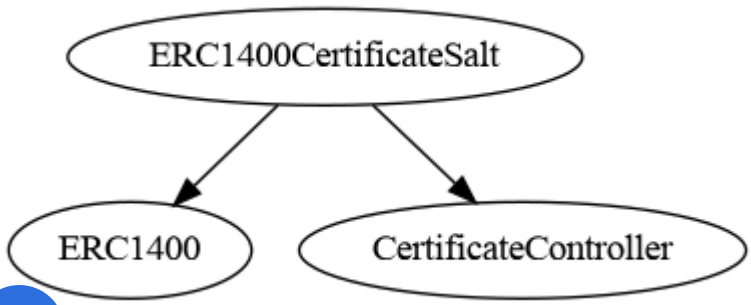
## ERC1400




## ERC1400CertificateNonce




## ERC1400CertificateNonce



## A.4.3 Tests Suite

The tests cover 100% of the code.

Below is the output generated by running the test suite:

```
yarn run v1.16.0
$ yarn truffle run coverage
$ truffle run coverage
⚠  Unable to require Truffle library locally or globally.
⚠  Using fallback Truffle library module instead (v5.0.31)
⚠  Truffle V5 must be a local dependency for fallback to work.

> server:            http://127.0.0.1:8555
> truffle:           v5.0.31
> ganache-core:      v2.5.7
> solidity-coverage: v0.7.0-beta.3

Network Info
============
> id:      *
> port:    8555
> network: soliditycoverage


Instrumenting for coverage...
=============================

> ERC1400.sol
> extensions/tokenExtensions/ERC1400TokensChecker.sol
> extensions/tokenExtensions/ERC1400TokensValidator.sol
> extensions/tokenExtensions/IERC1400TokensChecker.sol
> extensions/tokenExtensions/IERC1400TokensValidator.sol
> extensions/tokenExtensions/roles/BlacklistAdminRole.sol
> extensions/tokenExtensions/roles/BlacklistedRole.sol
> extensions/userExtensions/IERC1400TokensRecipient.sol
> extensions/userExtensions/IERC1400TokensSender.sol
> IERC1400.sol
> interface/ERC1820Implementer.sol
> mocks/ERC1400TokensRecipientMock.sol
> mocks/ERC1400TokensSenderMock.sol
> mocks/FakeERC1400Mock.sol
> tokens/ERC20Token.sol
> tokens/ERC721Token.sol
> tools/BatchBalanceReader.sol
> tools/BatchTokenIssuer.sol
> tools/DVP.sol
```

```
Coverage skipped for:
=====================

> certificate/certificateControllers/CertificateControllerNonce.sol
> certificate/certificateControllers/CertificateControllerSalt.sol
> certificate/ERC1400CertificateNonce.sol
> certificate/ERC1400CertificateSalt.sol
> Migrations.sol
> mocks/BlacklistMock.sol
> mocks/CertificateControllerMock.sol
> mocks/ERC1400CertificateMock.sol
> tools/FundIssuer.sol

Compiling your contracts...
===========================
> Compiling ./.coverage_contracts/ERC1400.sol
> Compiling ./.coverage_contracts/IERC1400.sol
> Compiling ./.coverage_contracts/Migrations.sol
> Compiling ./.coverage_contracts/certificate/ERC1400CertificateNonce.sol
> Compiling ./.coverage_contracts/certificate/ERC1400CertificateSalt.sol
> Compiling ./.coverage_contracts/certificate/certificateControllers/Certifi
> Compiling ./.coverage_contracts/certificate/certificateControllers/Certifi
> Compiling ./.coverage_contracts/extensions/tokenExtensions/ERC1400TokensCh
> Compiling ./.coverage_contracts/extensions/tokenExtensions/ERC1400TokensVa
> Compiling ./.coverage_contracts/extensions/tokenExtensions/IERC1400TokensC
> Compiling ./.coverage_contracts/extensions/tokenExtensions/IERC1400TokensV
> Compiling ./.coverage_contracts/extensions/tokenExtensions/roles/Blacklist
> Compiling ./.coverage_contracts/extensions/tokenExtensions/roles/Blacklist
> Compiling ./.coverage_contracts/extensions/userExtensions/IERC1400TokensRe
> Compiling ./.coverage_contracts/extensions/userExtensions/IERC1400TokensSe
> Compiling ./.coverage_contracts/interface/ERC1820Implementer.sol
> Compiling ./.coverage_contracts/mocks/BlacklistMock.sol
> Compiling ./.coverage_contracts/mocks/CertificateControllerMock.sol
> Compiling ./.coverage_contracts/mocks/ERC1400CertificateMock.sol
> Compiling ./.coverage_contracts/mocks/ERC1400TokensRecipientMock.sol
> Compiling ./.coverage_contracts/mocks/ERC1400TokensSenderMock.sol
> Compiling ./.coverage_contracts/mocks/FakeERC1400Mock.sol
> Compiling ./.coverage_contracts/tokens/ERC20Token.sol
> Compiling ./.coverage_contracts/tokens/ERC721Token.sol
> Compiling ./.coverage_contracts/tools/BatchBalanceReader.sol
> Compiling ./.coverage_contracts/tools/BatchTokenIssuer.sol
> Compiling ./.coverage_contracts/tools/DVP.sol
> Compiling ./.coverage_contracts/tools/FundIssuer.sol
> Compiling erc1820/contracts/ERC1820Client.sol
> Compiling openzeppelin-solidity/contracts/access/Roles.sol
> Compiling openzeppelin-solidity/contracts/access/roles/MinterRole.sol
> Compiling openzeppelin-solidity/contracts/access/roles/PauserRole.sol
> Compiling openzeppelin-solidity/contracts/access/roles/WhitelistAdminRole.
> Compiling openzeppelin-solidity/contracts/access/roles/WhitelistedRole.sol
```

```
> Compiling openzeppelin-solidity/contracts/introspection/ERC165.sol
> Compiling openzeppelin-solidity/contracts/introspection/IERC165.sol
> Compiling openzeppelin-solidity/contracts/lifecycle/Pausable.sol
> Compiling openzeppelin-solidity/contracts/math/SafeMath.sol
> Compiling openzeppelin-solidity/contracts/ownership/Ownable.sol
> Compiling openzeppelin-solidity/contracts/token/ERC20/ERC20.sol
> Compiling openzeppelin-solidity/contracts/token/ERC20/ERC20Mintable.sol
> Compiling openzeppelin-solidity/contracts/token/ERC20/IERC20.sol
> Compiling openzeppelin-solidity/contracts/token/ERC721/ERC721.sol
> Compiling openzeppelin-solidity/contracts/token/ERC721/ERC721Mintable.sol
> Compiling openzeppelin-solidity/contracts/token/ERC721/IERC721.sol
> Compiling openzeppelin-solidity/contracts/token/ERC721/IERC721Receiver.sol
> Compiling openzeppelin-solidity/contracts/utils/Address.sol
> Artifacts written to /home/daniel/Development/github.com/ConsenSys/codefi-
> Compiled successfully using:
   - solc: 0.5.10+commit.5a6ea5b1.Emscripten.clang


Compiling your contracts...
===========================
> Compiling ./.coverage_contracts/certificate/ERC1400CertificateNonce.sol
> Compiling ./.coverage_contracts/certificate/certificateControllers/Certifi

Account to load with ETH:  0x367E65148e058399729bd42490A9945997B2158d

   > ERC1820 deployment: Success --> 0x1820a4B7618BdE71Dce8cdc73aAB6C95905fa

   > ERC1400 token deployment: Success --> 0x33A68097450E8303cD012BcdC40FD51

   > ERC1400CertificateNonce token deployment: Success --> 0x3c5D013B0EC00DA

   > ERC1400CertificateSalt token deployment: Success --> 0x63A27d7458C7D410

   > Add token extension for token deployed at address 0x33A68097450E8303cD0

   > Token extension deployment: Success --> 0xca85B3484D69A3F64081b5548f6E9

   > Token connection to extension: Success

   > Balance Reader deployment: Success --> 0x25C8c850A8e961656F525631D9F094

   > Balance Reader registry in ERC1820: Success --> 0x25C8c850A8e961656F525

   > Batch issuer deployment: Success --> 0xEC3C92a8822f2acc4205B15c9dcb3681

   > Batch issuer registry in ERC1820: Success --> 0xEC3C92a8822f2acc4205B15

   > DVP deployment: Success --> 0xf3De6CA3E00ED61E20E443a389139C7e0854ffF4

   > DVP registry in ERC1820: Success --> 0xf3De6CA3E00ED61E20E443a389139C7e
```

```
 > FundIssuer deployment: Success --> 0xda4F2460DF89f220F88335181F674FC1d6

 > FundIssuer registry in ERC1820: Success --> 0xda4F2460DF89f220F88335181


Contract: BatchBalanceReader
  balancesOfByPartition
    ✓ returns the partition balances list (326ms)
  balancesOf
    ✓ returns the balances list (99ms)

Contract: BatchTokenIssuer
  batchIssueByPartition
    when input is correct
      when the operator is the owner of the token contract
        ✓ issues tokens for multiple different holders (4287ms)
      when the operator has been declared as minter in the BatchTokenIssue
        ✓ issues tokens for multiple different holders (4180ms)
      when the operator neither the owner of the token contract, nor a mir
        ✓ issues tokens for multiple different holders (1989ms)
    when tokenHoler list is not correct
      ✓ reverts (144ms)
    when values list is not correct
      ✓ reverts (179ms)
  setTokenMinters
    when the caller is the token contract owner
      ✓ sets the operators as token minters (188ms)
    when the caller is an other token minter
      ✓ sets the operators as token minters (312ms)
    when the caller is neither the token contract owner nor a token minter
      ✓ reverts (144ms)

Contract: ERC1400 with CertificateController
  setCertificateSigner
    when the sender is the contract owner
      when the new certificate signer address is valid
        ✓ sets the operator as certificate signer (148ms)
        ✓ sets the operator as certificate signer (283ms)
      when the certificate signer address is not valid
        ✓ reverts (128ms)
        ✓ reverts (98ms)
    when the sender is not the contract owner
      ✓ reverts (98ms)
  setCertificateControllerActivated
    when the sender is the contract owner
      ✓ deactivates the certificate controller (112ms)
      ✓ deactivates and reactivates the certificate controller (853ms)
    when the sender is not the contract owner
      ✓ reverts (88ms)
```

```
    transferByPartition
      when thecertifiacte is valid
        ✓ transfers the requested amount (385ms)
        ✓ emits a Checked event (220ms)
      when thecertifiacte is not valid
        ✓ reverts (111ms)

  Contract: DVP
    parameters
      owner
        ✓ returns the owner of the contract (236ms)
      tradeExecuters
        ✓ returns the list of trade executers (247ms)
        ✓ returns empty list of trade executers (222ms)
    canImplementInterfaceForAddress
      when the interface label is ERC777TokensRecipient
        ✓ returns ERC1820_ACCEPT_MAGIC
      when the interface label is not ERC777TokensRecipient
        ✓ returns empty bytes32
    canReceive
      when operatorData is not empty
        when data has the correct length
          when data has the right format
            when data is formatted for a trade proposal
              ✓ returns true [33m (41ms)
            when data is formatted for a trade acceptance
              ✓ returns true
          when data does not have the right format
            ✓ returns false
            ✓ returns false
        when data does not have the correct length
          ✓ returns false
      when operatorData is empty
        ✓ returns false
    tokensReceived
      when hook is called from ERC1400 contract
        when recipient is the DVP contract
          when data field is valid
            when received tokens correspond to a new trade proposal
              ✓ creates and accepts the trade request (564ms)
              ✓ creates and accepts a second trade request (910ms)
            when received tokens correspond to an existing trade acceptance
              when trade state is PENDING
                when trade recipient is defined
                  when token sender is the holder registered in the trade
                    when token is the correct token
                      when partition is the correct partition
                        when token standard for the trade is ERC1400
                          when token amount is correct
                            when there is an executer
```

```
                        ✓ accepts the trade request (1168ms)
                      when there is no executer
                        ✓ accepts and executes the trade request [USE
                    when token amount is not correct
                      when there is no executer
                        ✓ reverts (1046ms)
                    when token standard for the trade is not ERC1400
                      ✓ reverts (1020ms)
                  when partition is not the correct partition
                    ✓ reverts (989ms)
                when token is not the correct token
                  ✓ reverts (1059ms)
              when token sender is not the holder registered in the trac
                ✓ reverts (938ms)
            when trade recipient is not defined
              when there is an executer
                ✓ accepts and executes the trade request [USE CASE - ATC
          when trade state is not PENDING
            ✓ reverts (1650ms)
        when data field is not valid
          ✓ reverts (1013ms)
      when recipient is not the DVP contract
        ✓ reverts (337ms)
    when hook is not called from ERC1400 contract
      ✓ reverts (133ms)
  requestTrade
    when none of the 2 tokens is ETH
      when the DVP contract is not controllable
        when escrowable is not forbidden
          when expiration date is defined
            when sender is holder 1
              when DVP request is of type Escrow
                when token standard is ERC20
                  ✓ creates and accepts the trade request (609ms)
                when token standard is ERC721
                  ✓ creates and accepts the trade request (752ms)
                when token standard is ERC1400
                  ✓ creates and accepts the trade request (1251ms)
                when payment is made off-chain
                  ✓ creates and accepts the trade request (344ms)
              when DVP request is of type Swap
                when token standard is ERC20
                  ✓ creates and accepts the trade request (558ms)
                when token standard is ERC721
                  ✓ creates and accepts the trade request (762ms)
                when token standard is ERC1400
                  ✓ creates and accepts the trade request (1072ms)
                when payment is made off-chain
                  ✓ creates and accepts the trade request (838ms)
            when sender is holder 2
```

```
            when sender is holder 2
              ✓ creates and accepts the trade request (568ms)
            when sender is neither holder 1 nor holder 2
              when the holder 1 is not the zero address
                ✓ creates the trade request (345ms)
              when the holder 1 is the zero address
                ✓ reverts (123ms)
          when expiration date is not defined
            ✓ creates the trade request (583ms)
        when escrowable is forbidden
          when escrow mode is not requested
            ✓ creates the trade request (626ms)
          when escrow mode is requested
            ✓ reverts (388ms)
      when the DVP contract is owned
        when a valid trade executer is defined
          ✓ creates the trade request (550ms)
        when no valid trade executer is defined
          when proposed executer for the trade is not in the list of DVP t
            ✓ reverts (340ms)
          when proposed trade executer is zero address
            ✓ reverts (386ms)
    when one of the 2 tokens is ETH
      when proposed trade type is Escrow
        when sender is holder 1
          ✓ creates the trade request (367ms)
        when sender is holder 2
          ✓ creates the trade request (368ms)
      when proposed trade type is Swap
        ✓ creates the trade request (195ms)
  acceptTrade
    when trade index is valid
      when tokens need to be escrowed
        when tokens are available
          when trade has no predefined executer
            when there are no token controllers
              when trade gets executed
                ✓ accepts and executes the trade (1152ms)
              when trade doesnt get executed
                ✓ accepts the trade (864ms)
            when there are token controllers
              ✓ accepts the trade (1107ms)
          when trade has predefined executer
            ✓ accepts the trade (1035ms)
        when tokens are not available
          when token standard is ETH
            ✓ reverts (599ms)
          when token standard is ERC20
            ✓ reverts (900ms)
          when token standard is ERC1400
```

```
            ✓ reverts (875ms)
        when tokens do not need to be escrowed
          when token standard is ERC20
            when tokens have been reserved before
              ✓ accepts and executes the trade (1352ms)
            when tokens have not been reserved before
              ✓ reverts (918ms)
          when token standard is ERC721
            when tokens have been reserved before
              ✓ accepts and executes the trade (1595ms)
            when tokens have not been reserved before
              ✓ reverts (851ms)
          when token standard is ERC1400
            when tokens have been reserved before
              ✓ accepts and executes the trade (1388ms)
            when tokens have not been reserved before
              ✓ reverts (863ms)
          when payment is made off-chain
            ✓ accepts and executes the trade (1176ms)
      when trade index is not valid
        when trade with indicated index doesn t exist
          ✓ reverts (1017ms)
        when trade with indicated index is not in state pending
          ✓ reverts (1003ms)
    approveTrade
      when trade index is valid
        when sender is token controller
          when one single approval is required
            when trade is executed
              ✓ approves and executes the trade (1552ms)
            when trade is not executed
              ✓ approves the trade (1347ms)
              ✓ approves, disapproves and re-approves the trade (1824ms)
          when two approvals are required
            when trade is executed
              ✓ approves and executes the trade (2299ms)
            when trade is not executed
              ✓ approves the trade (1707ms)
        when sender is not token controller
          ✓ reverts (1332ms)
      when trade index is not valid
        when trade with indicated index doesn t exist
          ✓ reverts (1161ms)
        when trade with indicated index is not in state pending
          ✓ reverts (1495ms)
    executeTrade
      when trade index is valid
        when caller is executer defined at trade creation
          when trade has been approved
            when trade has been accepted
```

```
            when trade is executed at initially defined price
              when expiration date is not past
                when token standard is ERC20 vs ERC20
                  when trade type is Escrow
                    ✓ executes the trade (1347ms)
                  when trade type is Swap
                    when trade is executed by an executer
                      when tokens are available
                        ✓ executes the trade (1504ms)
                      when tokens are not available
                        ✓ executes the trade (1435ms)
                    when trade is executed by a holder
                      ✓ executes the trade (2042ms)
                when token standard is ERC20 vs ETH
                  when trade type is Escrow
                    ✓ executes the trade (1193ms)
                when token standard is ERC20 vs off-chain payment
                  when trade type is Escrow
                    ✓ executes the trade (1502ms)
                when token standard is ERC721 vs ERC20
                  when trade type is Escrow
                    ✓ executes the trade (1398ms)
                when token standard is ERC1400 vs ERC20
                  when trade type is Escrow
                    ✓ executes the trade (1806ms)
              when expiration date is past
                ✓ reverts (1423ms)
            when trade is not executed at initially defined price
              ✓ creates and accepts the trade request (1127ms)
          when trade has not been accepted
            ✓ reverts (822ms)
        when trade has not been approved
          ✓ reverts (1390ms)
      when caller is not executer defined at trade creation
        ✓ reverts (1258ms)
    when trade index is not valid
      ✓ reverts (112ms)
  forceTrade
    when trade index is valid
      when trade has not been accepted by both parties
        when traded tokens have no controllers
          when executer has not been defined at trade creation
            when trade has been accepted by holder1
              when sender is holder1
                ✓ forces the trade (879ms)
              when sender is not holder1
                ✓ reverts (784ms)
            when trade has been accepted by holder2
              when sender is holder2
                ✓ forces the trade (911ms)
```

```
            when sender is not holder2
              ✓ reverts (745ms)
          when trade has been accepted neither by holder1, nor by holder
            ✓ reverts (617ms)
        when executer has been defined at trade creation
          when caller is executer defined at trade creation
            ✓ executes the trade (815ms)
          when caller is not executer defined at trade creation
            ✓ executes the trade (678ms)
      when at least one of traded tokens has controllers
        ✓ reverts (763ms)
    when trade has been accepted by both parties
      ✓ reverts (1287ms)
  when trade index is not valid
    ✓ reverts (93ms)
cancelTrade
  when trade index is valid
    when trade has been accepted by both parties
      when caller is trade executer
        when trade type is Escrow
          ✓ cancels the trade (1432ms)
        when trade type is Swap
          ✓ cancels the trade (1364ms)
      when caller is holder1
        when expiration date is past
          ✓ cancels the trade (1332ms)
        when expiration date is not past
          ✓ reverts (1185ms)
      when caller is holder2
        when expiration date is past
          ✓ cancels the trade (1335ms)
        when expiration date is not past
          ✓ reverts (1287ms)
    when trade has been accepted by holder1
      when caller is trade executer
        when trade type is Escrow
          ✓ cancels the trade (760ms)
        when trade type is Swap
          ✓ cancels the trade (772ms)
      when caller is holder1
        when expiration date is past
          ✓ cancels the trade (828ms)
        when expiration date is not past
          ✓ reverts (714ms)
      when caller is holder2
        when expiration date is past
          ✓ cancels the trade (718ms)
        when expiration date is not past
          ✓ reverts (746ms)
    when trade has been accepted by holder2
```

```
        when caller is trade executer
          when trade type is Escrow
            ✓ cancels the trade (1230ms)
          when trade type is Swap
            ✓ cancels the trade (1487ms)
        when caller is holder1
          when expiration date is past
            ✓ cancels the trade (1080ms)
          when expiration date is not past
            ✓ reverts (1086ms)
        when caller is holder2
          when expiration date is past
            ✓ cancels the trade (1150ms)
          when expiration date is not past
            ✓ reverts (1068ms)
      when trade has been accepted by no one
        when caller is trade executer
          ✓ cancels the trade (650ms)
        when caller is holder1
          ✓ cancels the trade (599ms)
        when caller is holder2
          ✓ cancels the trade (588ms)
        when caller is neither the executer nor one of the 2 holders
          ✓ cancels the trade (472ms)
    when trade index is not valid
      ✓ reverts (87ms)
  renounceOwnership
    when the caller is the contract owner
      ✓ renounces to ownership (336ms)
    when the caller is not the contract owner
      ✓ reverts (96ms)
  setTradeExecuters
    when the caller is the contract owner
      when the dvp contract is owned
        ✓ sets the operators as trade executers (193ms)
      when the dvp contract is not owned
        ✓ reverts (109ms)
    when the caller is not the contract owner
      ✓ reverts (95ms)
  setTokenControllers
    when the caller is the token contract owner
      ✓ sets the operators as token controllers (163ms)
    when the caller is an other token controller
      ✓ sets the operators as token controllers (298ms)
    when the caller is neither the token contract owner nor a token contro
      ✓ reverts (115ms)
  setPriceOracles
    when the caller is the token contract owner
      ✓ sets the operators as token price oracle (142ms)
    when the caller is an other price oracle
```

```
      when the caller is an other price oracle
        ✓ sets the operators as token price oracle (296ms)
      when the caller is neither the token contract owner nor a token price
        ✓ reverts (150ms)
    setPriceOwnership
      when sender is price oracle of the token
        ✓ takes the price ownership for a given token (250ms)
      when sender is not price oracle of the token
        ✓ reverts (102ms)
    setTokenPrice
      when there is no competition on the price ownership
        when the price ownership is taken
          when the price ownership is taken by the right person
            ✓ sets the price for token1 (316ms)
            ✓ sets the price for token2 (289ms)
          when the price ownership is not taken by the right person
            ✓ reverts (237ms)
            ✓ reverts (246ms)
        when the price ownership is not taken
          ✓ sets the price for token1 (101ms)
      when there is competition on the price ownership
        ✓ reverts (110ms)
    setVariablePriceStartDate
      when sender is price oracle of the token
        when start date is further than a week
          ✓ sets the variable price start date for a given token (260ms)
        when start date is not further than a week
          ✓ reverts (121ms)
      when sender is not price oracle of the token
        ✓ reverts (131ms)
    getPrice
      when the variable price start date has been set
        when there is no competition on the price ownership
          when the price ownership is taken
            when the first token has more value than the second token
              when the price ownership is taken for the first token
                when the price is set (case 1)
                  ✓ returns the updatedprice [33m (61ms)
                  when the price is set (case 2)
                    ✓ returns the updatedprice [33m (51ms)
                    when the price is set (case 3)
                      ✓ returns the updatedprice [33m (52ms)
                      when the price is set (case 4)
                        ✓ returns the updatedprice [33m (51ms)
                        ✓ executes the trade at correct price (840ms)
              when the price is not set
                ✓ returns the price defined in the trade [33m (62ms)
            when the price ownership is taken for the second token
              when the price is set (case 1)
                ✓ returns the updatedprice [33m (67ms)
```

```
            when the second token has more value than the first token
              when the price ownership is taken for the first token
                when the price is set (case 1)
                  ✓ returns the updatedprice [33m (75ms)
                  when the price is set (case 2)
                    ✓ returns the updatedprice [33m (61ms)
                    when the price is set (case 3)
                      ✓ returns the updatedprice [33m (63ms)
                      when the price is set (case 4)
                        ✓ returns the updatedprice [33m (54ms)
                        ✓ reverts when price is higher than amount escrowed
                when the price is not set
                  ✓ returns the price defined in the trade [33m (72ms)
              when the price ownership is taken for the second token
                when the price is set (case 1)
                  ✓ returns the updatedprice [33m (74ms)
          when the price ownership is not taken
            ✓ returns the price defined in the trade [33m (39ms)
        when there is competition on the price ownership
          ✓ reverts [33m (52ms)
      when the variable price start date has been set
        ✓ returns the non-updated price (238ms)

  Contract: ERC1400
    contract creation
      ✓ fails deploying the contract if granularity is lower than 1 (314ms)
    canImplementInterfaceForAddress
      when interface hash is correct
        ✓ returns ERC1820_ACCEPT_MAGIC [33m (50ms)
      when interface hash is not correct
        ✓ returns ERC1820_ACCEPT_MAGIC
    transfer
      when the amount is a multiple of the granularity
        when the recipient is not the zero address
          when the sender has enough balance
            ✓ transfers the requested amount (257ms)
            ✓ emits a Transfer event (216ms)
          when the sender does not have enough balance
            ✓ reverts (421ms)
        when the recipient is the zero address
          ✓ reverts (241ms)
      when the amount is not a multiple of the granularity
        ✓ reverts (702ms)
    transferFrom
      when token has a withelist
        when the operator is approved
          when the amount is a multiple of the granularity
            when the recipient is not the zero address
              when the sender has enough balance
                ✓ transfers the requested amount (249ms)
```

```
            ✓ emits a sent + a transfer event (196ms)
          when the sender does not have enough balance
            ✓ reverts (106ms)
        when the recipient is the zero address
          ✓ reverts (304ms)
      when the amount is not a multiple of the granularity
        ✓ reverts (560ms)
    when the operator is not approved
      when the operator is not approved but authorized
        ✓ transfers the requested amount (371ms)
      when the operator is not approved and not authorized
        ✓ reverts (111ms)
  approve
    when sender approves an operator
      ✓ approves the operator (148ms)
      ✓ emits an approval event (111ms)
    when the operator to approve is the zero address
      ✓ reverts (125ms)
  set/getDocument
    setDocument
      when sender is a controller
        ✓ attaches the document to the token (168ms)
        ✓ emits a document event (130ms)
      when sender is not a controller
        ✓ reverts (109ms)
    getDocument
      when docuemnt exists
        ✓ returns the document (190ms)
      when docuemnt does not exist
        ✓ reverts [33m (38ms)
  partitionsOf
    when tokenHolder owes no tokens
      ✓ returns empty list
    when tokenHolder owes tokens of 1 partition
      ✓ returns partition (207ms)
    when tokenHolder owes tokens of 3 partitions
      ✓ returns list of 3 partitions (536ms)
  transferWithData
    when defaultPartitions have been defined
      when the amount is a multiple of the granularity
        when the recipient is not the zero address
          when the sender has enough balance for those default partitions
            when the sender has defined custom default partitions
              ✓ transfers the requested amount (821ms)
              ✓ emits a sent event (576ms)
            when the sender has not defined custom default partitions
              ✓ transfers the requested amount (672ms)
          when the sender does not have enough balance for those default p
            ✓ reverts (1063ms)
        when the recipient is the zero address
```

```
        ✓ reverts (431ms)
      when the amount is not a multiple of the granularity
        ✓ reverts (1282ms)
    when defaultPartitions have not been defined
      ✓ reverts (908ms)
  transferFromWithData
    when the operator is approved
      when the amount is a multiple of the granularity
        when the recipient is not the zero address
          when defaultPartitions have been defined
            when the sender has enough balance for those default partitior
              ✓ transfers the requested amount (819ms)
              ✓ emits a sent event (581ms)
            when the sender does not have enough balance for those default
              ✓ reverts (1028ms)
              ✓ reverts (mock contract - for 100% test coverage) (628ms)
          when defaultPartitions have not been defined
            ✓ reverts (251ms)
        when the recipient is the zero address
          ✓ reverts (425ms)
      when the amount is not a multiple of the granularity
        ✓ reverts (1095ms)
    when the operator is not approved
      ✓ reverts (244ms)
  transferByPartition
    when the sender has enough balance for this partition
      when the transfer amount is not equal to 0
        ✓ transfers the requested amount (504ms)
        ✓ emits a TransferByPartition event (195ms)
      when the transfer amount is equal to 0
        ✓ reverts (163ms)
    when the sender does not have enough balance for this partition
      ✓ reverts (106ms)
  operatorTransferByPartition
    when the sender is approved for this partition
      when approved amount is sufficient
        ✓ transfers the requested amount (540ms)
      when approved amount is not sufficient
        ✓ reverts (353ms)
    when the sender is an operator for this partition
      when the sender has enough balance for this partition
        when partition does not change
          ✓ transfers the requested amount (496ms)
          ✓ transfers the requested amount with attached data (without cha
          ✓ emits a TransferByPartition event (287ms)
        when partition changes
          ✓ transfers the requested amount (540ms)
          ✓ converts the requested amount (427ms)
          ✓ emits a changedPartition event (309ms)
      when the sender does not have enough balance for this partition
```

```
      ✓ reverts (226ms)
    when the sender is a global operator
      ✓ redeems the requested amount (439ms)
    when the sender is neither an operator, nor approved
      ✓ reverts (143ms)
  authorizeOperator
    when sender authorizes an operator
      ✓ authorizes the operator (174ms)
      ✓ emits a authorized event (144ms)
    when sender authorizes himself
      ✓ reverts (105ms)
  revokeOperator
    when sender revokes an operator
      ✓ revokes the operator (when operator is not the controller) (315ms)
      ✓ emits a revoked event (101ms)
    when sender revokes himself
      ✓ reverts (108ms)
  authorizeOperatorByPartition
    ✓ authorizes the operator (146ms)
    ✓ emits an authorized event (96ms)
  revokeOperatorByPartition
    when operator is not controller
      ✓ revokes the operator (236ms)
      ✓ emits a revoked event (196ms)
  isOperator
    ✓ when operator is tokenHolder
    ✓ when operator is authorized by tokenHolder (129ms)
    ✓ when is a revoked operator (224ms)
    ✓ when is a controller and token is controllable
    ✓ when is a controller and token is not controllable (134ms)
  isOperatorForPartition
    ✓ when operator is tokenHolder
    ✓ when operator is authorized by tokenHolder (121ms)
    ✓ when is a revoked operator (244ms)
    ✓ when is a controller and token is controllable
    ✓ when is a controller and token is not controllable (117ms)
  issue
    when sender is the issuer
      when token is issuable
        when default partitions have been defined
          when the amount is a multiple of the granularity
            when the recipient is not the zero address
              ✓ issues the requested amount (198ms)
              ✓ issues twice the requested amount (392ms)
              ✓ emits a issuedByPartition event (177ms)
            when the recipient is not the zero address
              ✓ issues the requested amount (139ms)
          when the amount is not a multiple of the granularity
            ✓ issues the requested amount (458ms)
        when default partitions have not been defined
```

```
          when default partitions have not been defined
            ✓ reverts (415ms)
        when token is not issuable
          ✓ reverts (249ms)
      when sender is not the issuer
        ✓ reverts (98ms)
    issueByPartition
      when sender is the issuer
        when token is issuable
          ✓ issues the requested amount (259ms)
          ✓ issues twice the requested amount (379ms)
          ✓ emits a issuedByPartition event (174ms)
        when token is not issuable
          ✓ reverts (280ms)
      when sender is not the issuer
        ✓ reverts (125ms)
    redeem
      when defaultPartitions have been defined
        when the amount is a multiple of the granularity
          when the sender has enough balance for those default partitions
            ✓ redeeems the requested amount (583ms)
            ✓ emits a redeemedByPartition events (465ms)
          when the sender does not have enough balance for those default par
            ✓ reverts (832ms)
        when the amount is not a multiple of the granularity
          ✓ reverts (1131ms)
      when defaultPartitions have not been defined
        ✓ reverts (213ms)
    redeemFrom
      when the operator is approved
        when defaultPartitions have been defined
          when the sender has enough balance for those default partitions
            when the amount is a multiple of the granularity
              when the redeemer is not the zero address
                ✓ redeems the requested amount (629ms)
                ✓ emits redeemedByPartition events (483ms)
              when the redeemer is the zero address
                ✓ reverts (363ms)
                ✓ reverts (mock contract - for 100% test coverage) (1184ms)
            when the amount is not a multiple of the granularity
              ✓ reverts (1162ms)
          when the sender does not have enough balance for those default par
            ✓ reverts (800ms)
            ✓ reverts (mock contract - for 100% test coverage) (1051ms)
        when defaultPartitions have not been defined
          ✓ reverts (208ms)
      when the operator is not approved
        ✓ reverts (212ms)
    redeemByPartition
      when the redeemer has enough balance for this partition
        ✓ redeems the requested amount (226ms)
```

```
      ✓ redeems the requested amount (226ms)
      ✓ emits a redeemedByPartition event (182ms)
    when the redeemer does not have enough balance for this partition
      ✓ reverts (115ms)
    special case (_removeTokenFromPartition shall revert)
      ✓ reverts (337ms)
  operatorRedeemByPartition
    when the sender is an operator for this partition
      when the redeemer has enough balance for this partition
        ✓ redeems the requested amount (341ms)
        ✓ emits a redeemedByPartition event (254ms)
      when the redeemer does not have enough balance for this partition
        ✓ reverts
    when the sender is a global operator
      ✓ redeems the requested amount (305ms)
    when the sender is not an operator
      ✓ reverts (129ms)
  parameters
    name
      ✓ returns the name of the token
    symbol
      ✓ returns the symbol of the token
    decimals
      ✓ returns the decimals the token
    granularity
      ✓ returns the granularity of tokens
    totalPartitions
      ✓ returns the list of partitions (564ms)
    total supply
      ✓ returns the total amount of tokens (191ms)
    balanceOf
      when the requested account has no tokens
        ✓ returns zero
      when the requested account has some tokens
        ✓ returns the total amount of tokens (184ms)
    controllers
      ✓ returns the list of controllers
    implementer1400
      ✓ returns the contract address
    implementer20
      ✓ returns the zero address
  setControllers
    when the caller is the contract owner
      ✓ sets the operators as controllers (478ms)
    when the caller is not the contract owner
      ✓ reverts (94ms)
  setPartitionControllers
    when the caller is the contract owner
      ✓ sets the operators as controllers for the specified partition (556
      ✓ removes the operators as controllers for the specified partition (
```

```
      when the caller is not the contract owner
        ✓ reverts (133ms)
    defaultPartitions
      when the sender is the contract owner
        ✓ sets the list of token default partitions (173ms)
      when the sender is not the contract owner
        ✓ reverts (120ms)
    approveByPartition
      when sender approves an operator for a given partition
        ✓ approves the operator (147ms)
        ✓ emits an approval event (99ms)
      when the operator to approve is the zero address
        ✓ reverts (90ms)
    migrate
      when the sender is the contract owner
        when the contract is not migrated
          ✓ can transfer tokens (322ms)
        when the contract is migrated definitely
          ✓ can not transfer tokens (481ms)
        when the contract is migrated, but not definitely
          ✓ can transfer tokens (578ms)
      when the sender is not the contract owner
        ✓ reverts (89ms)

  Contract: Fund issuance
    executePaymentAsInvestor
      when function is called by the investor
        when payment is made with ether
          when asset value is of type Unknown
            when cycle is at least in payment period
              when order is of type amount
                when asset value is not nil
                  when payment is not bypassed
                    when payment value is correct
                      when order state is Subscribed
                        ✓ updates the order state to Paid (205ms)
                      when order state is UnpaidSettled
                        when cycle is not finalized
                          ✓ reverts
                        when cycle is finalized
                          ✓ reverts
                      when order state is neither Subscribed nor UnpaidSettl
                        ✓ reverts
                    when payment value is not correct
                      ✓ reverts
                  when payment is bypassed
                    ✓ reverts
                when reverse asset value is not nil
                  ✓ reverts
              when order is of type value
```

```
            when asset value is not nil
              ✓ reverts
            when reverse asset value is not nil
              ✓ reverts
        when cycle is not at least in payment period
          ✓ reverts
      when asset value is of type Known
        when cycle is at least in subscription period
          ✓ reverts
        when cycle is not at least in subscription period
          ✓ reverts
    when payment is made with erc20
      when payment value is correct
      ✓ reverts
      when payment value is not correct
      ✓ reverts
    when payment is made with erc1400 through allowance
      when payment value is correct
      ✓ reverts
      when payment value is not correct
      ✓ reverts
    when payment is made with erc1400 through hook
      when payment value is correct
        when payment succeeds
          ✓ reverts
        when payment type is not correct
          ✓ reverts
        when payment address is not correct
          ✓ reverts
        when payment partition is not correct
          ✓ reverts
      when payment value is not correct
        ✓ reverts
    when payment is done off-chain
      ✓ reverts
  when function is not called by the investor
    ✓ reverts
rejectOrder
  when order exists and can still be rejected
    when valuation period is not over
      when message sender is the token controller
        when the order has not been settled
          when the order needs to be rejected
            when order has not been paid yet
              when we are in the subscription period
                ✓ rejects the order (190ms)
              when we are in the valuation period
                ✓ rejects the order (262ms)
        when the order rejection needs to be cancelled
          ✓ cancels the rejection (310ms)
```

```
          when the order has been settled
            when the order has been paid
              ✓ reverts
            when the order has not been paid
              ✓ reverts
        when message sender is not the token controller
          ✓ reverts
      when subscription period is over
        ✓ reverts
    when order can not be rejected
      when order doesnt exist
        ✓ reverts
      when order has already been settled
        when order has been paid
          ✓ reverts
        when order has not been paid
          ✓ reverts
      when order has been cancelled
        ✓ reverts
      when order has already been rejected
        ✓ reverts
  parameters
    implementerFund
      ✓ returns the contract address
  canImplementInterfaceForAddress
    when interface hash is correct
      ✓ returns ERC1820_ACCEPT_MAGIC
    when interface hash is not correct
      ✓ returns empty bytes32
  canReceive
    when operatorData is not empty
      when data has the correct length
        when data has the right format
          when data is formatted for an order creation
            ✓ returns true
          when data is formatted for an order payment
            ✓ returns true
          when data is formatted for a hook bypass
            ✓ returns true
        when data does not have the right format
          ✓ returns false
      when data does not have the correct length
        ✓ returns false
    when operatorData is empty
      ✓ returns false [33m (40ms)
  setAssetRules
    when caller is the token controller
      when first start time is valid
        when periods are valid
          when rules are not already defined
```

```
            ✓ sets asset rules (186ms)
          when rules are already defined
            ✓ updates asset rules (678ms)
        when periods are not valid
          when subscriptionPeriodLength is nil
            ✓ reverts (140ms)
          when valuationPeriodLength is nil
            ✓ reverts (125ms)
          when paymentPeriodLength is nil
            ✓ reverts (115ms)
      when first start time is not valid
        ✓ reverts (129ms)
    when caller is not the token controller
      ✓ reverts (136ms)
  subscribe
    when the current cycle is in subscription period
      when the current period is correct
        when order is of type value
          when value is not nil
            when asset value is unknown
              ✓ creates 2 new orders (852ms)
          when value is nil
            ✓ reverts (446ms)
        when order is of type amount
          when amount is not nil
            ✓ creates a new order (566ms)
          when amount is nil
            ✓ reverts (454ms)
      when the current period is not a subscription period (before first s
        ✓ reverts (430ms)
    when the current cycle is not in subscription period
      when rules are defined for the asset
        when subscriptions are open
          when cycle is the first cycle for this asset
            ✓ creates a new order (503ms)
            ✓ creates 3 orders (1688ms)
          when cycle is not the first cycle for this asset
            ✓ creates 3 orders (968ms)
        when subscriptions are not open
          ✓ reverts (340ms)
      when rules are not defined for the asset
        ✓ reverts (178ms)
  cancelOrder
    when order exists and can still be cancelled
      when subscription period is not over
        when message sender is the investor
          when order has not been paid yet
            ✓ cancels the order (214ms)
        when message sender is not the investor
          ✓ reverts (123ms)
```

```
            ✓ reverts (123ms)
        when subscription period is over
          ✓ reverts (208ms)
      when order can not be rejected
        when order doesnt exist
          ✓ reverts (97ms)
        when order has already been settled
          when order has been paid
            ✓ reverts
          when order has not been paid
            ✓ reverts
        when order has been cancelled
          ✓ reverts
        when order has already been rejected
          ✓ reverts
    valuate
      when we are in the valuation period
        when cycle is of type unknown
          when the provided values are valid
            when the sender is a price oracle
              ✓ sets the valuation (159ms)
              ✓ sets the reverse valuation (164ms)
              ✓ sets the valuation twice (292ms)
            when the sender is not a price oracle
              ✓ reverts (161ms)
          when the provided values are not valid
            ✓ reverts (134ms)
        when cycle is of type known
          ✓ set the valuation (1134ms)
      when we are in the subscription period
        ✓ reverts (146ms)
      when we are in the payment period
        ✓ reverts (137ms)

  Contract: ERC1400 with validator hook
    setHookContract
      when the caller is the contract owner
        ✓ sets the validator hook (161ms)
      when the caller is not the contract owner
        ✓ reverts (88ms)
    hooks
      when the transfer is successfull
        ✓ transfers the requested amount (301ms)
      when the transfer fails
        ✓ sender hook reverts (197ms)
    addBlacklisted/renounceBlacklistAdmin
      add/remove a blacklist admin
        when caller is a blacklist admin
          ✓ adds a blacklist admin (137ms)
          ✓ renounces blacklist admin (245ms)
```

```
      when caller is not a blacklist admin
        ✓ reverts (126ms)
  onlyNotBlacklisted
    can not call function if blacklisted
      ✓ reverts (329ms)
  whitelist
    can still call ERC1400 functions
      can still call issueByPartition
        ✓ issues new tokens (218ms)
      can still call redeemByPartition
        ✓ redeems the requested amount (254ms)
      can still call operatorRedeemByPartition
        ✓ redeems the requested amount (357ms)
      can still call transferByPartition
        ✓ transfers the requested amount (386ms)
      can still call operatorTransferByPartition
        ✓ transfers the requested amount (572ms)
      can still call redeem
        ✓ redeeems the requested amount (309ms)
      can still call redeemFrom
        ✓ redeems the requested amount (406ms)
      can still call transferWithData
        ✓ transfers the requested amount (305ms)
      can still call transferFromWithData
        ✓ transfers the requested amount (493ms)
    can not call ERC20 functions
      can still call transferWithData
        ✓ transfers the requested amount (278ms)
      can still call transferFromWithData
        ✓ transfers the requested amount (367ms)
  setWhitelistActivated
    when the caller is the contract owner
      ✓ activates the whitelist (389ms)
    when the caller is not the contract owner
      ✓ reverts (122ms)
  setBlacklistActivated
    when the caller is the contract owner
      ✓ activates the whitelist (351ms)
    when the caller is not the contract owner
      ✓ reverts (84ms)
  canTransferByPartition/canOperatorTransferByPartition
    when certificate is valid
      when checker has been setup
        when the operator is authorized
          when balance is sufficient
            when receiver is not the zero address
              when sender is eligible
                when validator is ok
                  when receiver is eligible
                    when the amount is a multiple of the granularity
```

```
                    ✓ returns Ethereum status code 51 (canTransferByPart
                    ✓ returns Ethereum status code 51 (canOperatorTrans
                  when the amount is not a multiple of the granularity
                    ✓ returns Ethereum status code 50 (168ms)
                when receiver is not eligible
                  ✓ returns Ethereum status code 57 (130ms)
              when validator is not ok
                ✓ returns Ethereum status code 54 (canTransferByPartitio
            when sender is not eligible
              ✓ returns Ethereum status code 56 (107ms)
          when receiver is the zero address
            ✓ returns Ethereum status code 57 (90ms)
        when balance is not sufficient
          ✓ returns Ethereum status code 52 (insuficient global balance)
          ✓ returns Ethereum status code 52 (insuficient partition balan
      when the operator is not authorized
        ✓ returns Ethereum status code 58 (canOperatorTransferByPartitio
    when checker has not been setup
      ✓ returns empty Ethereum status code 00 (canTransferByPartition)
  when certificate is not valid
    ✓ returns Ethereum status code 54 (canTransferByPartition)
    ✓ returns Ethereum status code 54 (canOperatorTransferByPartition)
whitelist/blacklist
  when token has a withlist
    when the sender and the recipient are whitelisted
      ✓ transfers the requested amount (358ms)
    when the sender is not whitelisted
      ✓ reverts (214ms)
    when the recipient is not whitelisted
      ✓ reverts (225ms)
  when token has a blacklist
    when the blacklist is activated
      when both the sender and the recipient are blacklisted
        ✓ reverts (722ms)
      when the sender is blacklisted
        ✓ reverts (367ms)
      when the recipient is blacklisted
        ✓ reverts (332ms)
      when neither the sender nor the recipient are blacklisted
        ✓ transfers the requested amount (523ms)
    when the blacklist is not activated
      when both the sender and the recipient are blacklisted
        ✓ transfers the requested amount (312ms)
  when token has neither a whitelist, nor a blacklist
    ✓ transfers the requested amount (265ms)
transferFrom
  when token has a withelist
    when the sender and the recipient are whitelisted
      when the operator is approved
        when the amount is a multiple of the granularity
```

```
              when the recipient is not the zero address
                when the sender has enough balance
                  ✓ transfers the requested amount (282ms)
                  ✓ emits a sent + a transfer event (236ms)
                when the sender does not have enough balance
                  ✓ reverts (108ms)
              when the recipient is the zero address
                ✓ reverts (286ms)
            when the amount is not a multiple of the granularity
              ✓ reverts (529ms)
          when the operator is not approved
            when the operator is not approved but authorized
              ✓ transfers the requested amount (402ms)
            when the operator is not approved and not authorized
              ✓ reverts (109ms)
        when the sender is not whitelisted
          ✓ reverts (99ms)
        when the recipient is not whitelisted
          ✓ reverts (100ms)
    pausable
      when contract is not paused
        ✓ transfers the requested amount (234ms)
        ✓ transfers the requested amount (509ms)
      when contract is paused
        ✓ reverts (201ms)
        ✓ reverts (237ms)

  Contract: ERC1400 with sender and recipient hooks
    hooks
      when the transfer is successfull
        ✓ transfers the requested amount (307ms)
      when the transfer fails
        ✓ sender hook reverts (186ms)
        ✓ recipient hook reverts (427ms)


 [92m  451 passing (10m)

-----------------------------------------------|----------|----------|--------
File                                           |  % Stmts | % Branch |  % Func
-----------------------------------------------|----------|----------|--------
 contracts/                                    |      100 |      100 |      10
  ERC1400.sol                                  |      100 |      100 |      10
  IERC1400.sol                                 |      100 |      100 |      10
 contracts/extensions/tokenExtensions/         |      100 |      100 |      10
  ERC1400TokensChecker.sol                     |      100 |      100 |      10
  ERC1400TokensValidator.sol                   |      100 |      100 |      10
  IERC1400TokensChecker.sol                    |      100 |      100 |      10
  IERC1400TokensValidator.sol                  |      100 |      100 |      10
 contracts/extensions/tokenExtensions/roles/   |      100 |      100 |      10
```

```
 BlacklistAdminRole.sol                    |      100 |      100 |      10
 BlacklistedRole.sol                       |      100 |      100 |      10
contracts/extensions/userExtensions/       |      100 |      100 |      10
 IERC1400TokensRecipient.sol               |      100 |      100 |      10
 IERC1400TokensSender.sol                  |      100 |      100 |      10
contracts/interface/                       |      100 |      100 |      10
 ERC1820Implementer.sol                    |      100 |      100 |      10
contracts/mocks/                           |      100 |      100 |      10
 ERC1400TokensRecipientMock.sol            |      100 |      100 |      10
 ERC1400TokensSenderMock.sol               |      100 |      100 |      10
 FakeERC1400Mock.sol                       |      100 |      100 |      10
contracts/tokens/                          |      100 |      100 |      10
 ERC20Token.sol                            |      100 |      100 |      10
 ERC721Token.sol                           |      100 |      100 |      10
contracts/tools/                           |      100 |      100 |      10
 BatchBalanceReader.sol                    |      100 |      100 |      10
 BatchTokenIssuer.sol                      |      100 |      100 |      10
 DVP.sol                                   |      100 |      100 |      10
-------------------------------------------|----------|----------|--------
All files                                  |      100 |      100 |      10
-------------------------------------------|----------|----------|--------

> Istanbul reports written to ./coverage/ and ./coverage.json
> solidity-coverage cleaning up, shutting down ganache server
Done in 661.14s.
```


# Request a Security Review Today

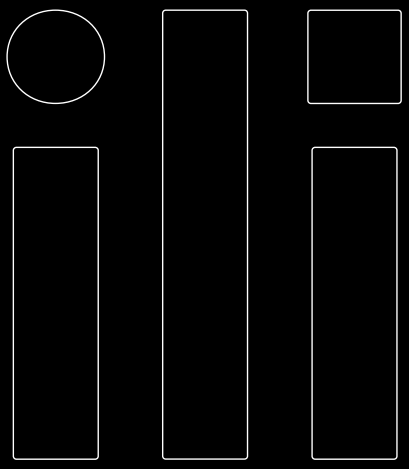
Get in touch with our team to request a quote for a smart contract audit.

CONTACT US

AUDITS

FUZZING

Subscribe to Our



FUZZING
SCRIBBLE
BLOG
TOOLS
RESEARCH
ABOUT
CONTACT
CAREERS
PRIVACY POLICY

## Newsletter

Stay up-to-date on our latest offerings, tools, and the world of blockchain security.

Email*

e-mail address

→

POWERED BY CONSENSYS