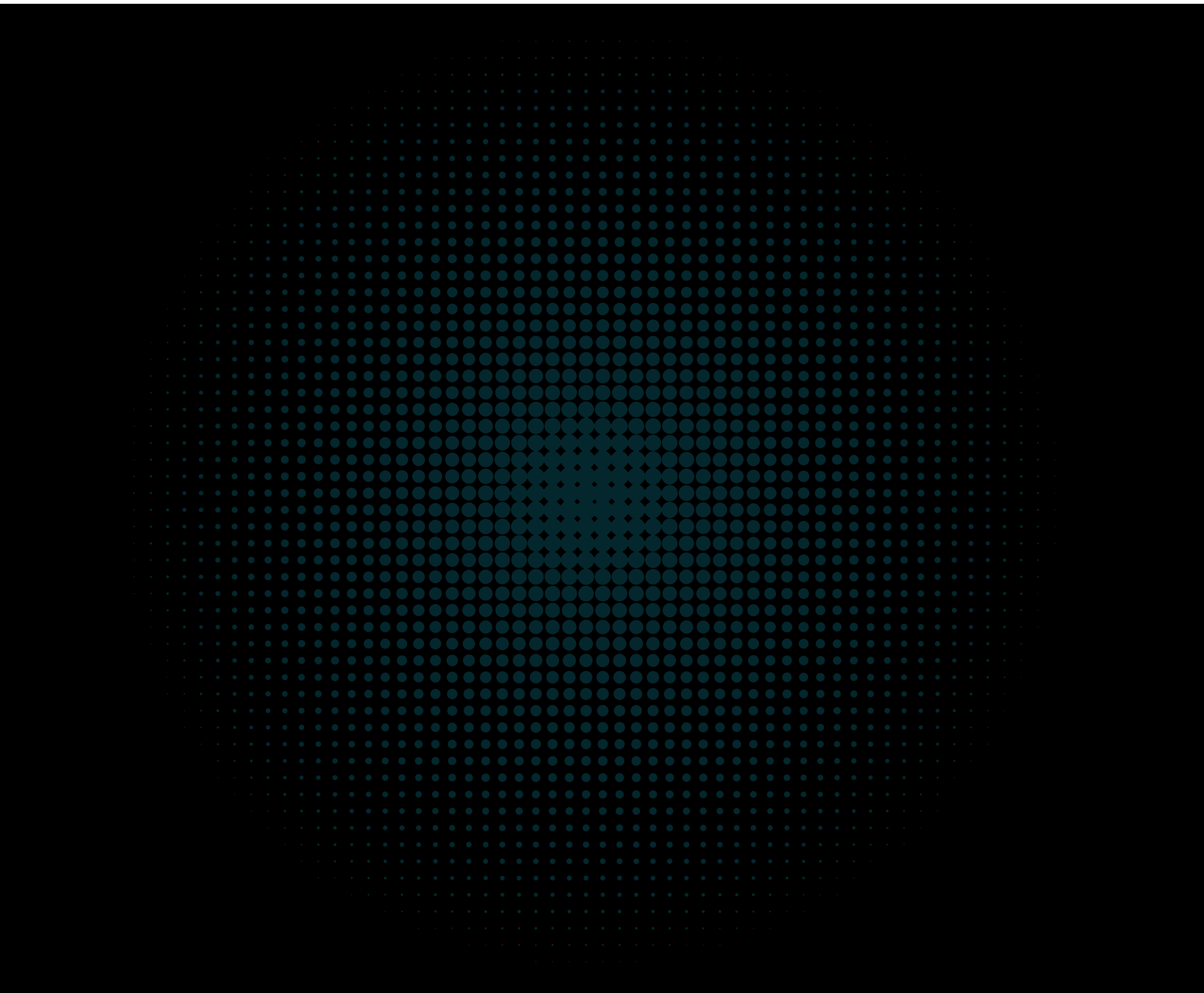
cmta.

RuleEngine Specification

Functional specifications of the RuleEngine for CMTAT and ERC-3643 tokens on Ethereum and EVM compatible blockchain.

RuleEngine version: v3.0.0-rc0
First published: August 2025



RuleEngine

This repository includes the RuleEngine contract for CMTAT and ERC-3643 tokens.

The RuleEngine is an external contract used to apply transfer restrictions to another contract, such as CMTAT and ERC-3643 tokens. Acting as a controller, it can call different contract rules and apply these rules on each transfer.

Intellectual property

Motivation

- Why use a dedicated contract with rules instead of implementing it directly in CMTAT or ERC-3643 tokens?

There are several reasons to do this:

- Flexibility: These different features are not standard and common to all tokens. From an implementation perspective, using a rule engine with custom rules allows for each issuer or contract user to decide which rules to apply.
- Code efficiency: The CMTAT token (and generally also all ERC-3643 tokens) is currently "heavy," meaning its contract code size is close to the maximum limit. This makes it challenging to add new features directly inside the token contract.
- Reusability:
  - We can use the RuleEngine inside other contracts besides CMTAT. For instance, the RuleEngine has been used it in our contract to distribute dividends (IncomeVault).
  - A same deployed `RuleEngine` can also be used with several different tokens if the rules allowed it, which is the case for all ready-only rule.

Why use this `RuleEngine` contract instead of setting directly the `rule` in the token contract?

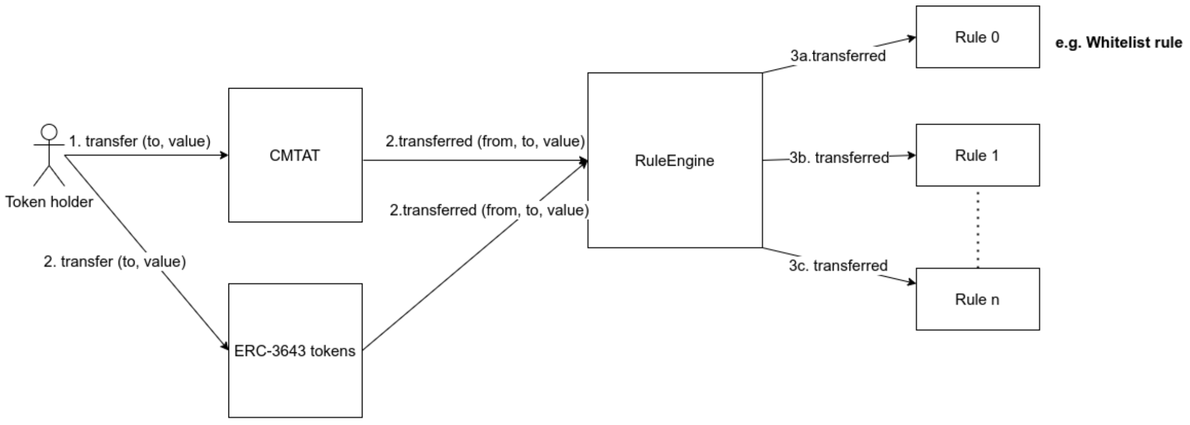
- Using a RuleEngine allows to call several different rules. For example, a blacklist rule to allow the issuer to manage its own list of blacklisted addresses and a sanctionlist rule to use the Chainalysis oracle for sanctions screening to forbid transfers from addresses listed in sanctions designations by organizations such as the US, EU, or UN.

When the use of `RuleEngine` may not be appropriate?

- If you plan to call only one rule (e.g a whitelist rule), it could make sense to directly set the rule in the token contract instead of using a RuleEngine. This will simplify configuration and reduce runtime gas costs.

How it works

This diagram illustrates how a transfer with a CMTAT or ERC-3643 token with a RuleEngine works:



1. The token holders initiate a transfer transaction on the token contract.
2. The transfer function inside the token calls the ERC-3643 function `transferred` from the RuleEngine with the following parameters inside: `from, to, value`.
3. The Rule Engine calls each rule separately. If the transfer is not authorized by the rule, the rule must directly revert (no return value).

How to set it

CMTAT v3.0.0

CMTAT provides the following function to set a RuleEngine inside a CMTAT token:

```
setRuleEngine(IRuleEngine ruleEngine_)
```

This function is defined in the extension module `ValidationModuleRuleEngine`

ERC-3643 token

ERC-3643 defined the following function in the standard interface to set a compliance contract

```
setCompliance(address _compliance)
```

How to include it

While the RuleEngine has been designed for CMTAT and ERC-3643 tokens, it can be used with other contracts to apply transfer restrictions.

For that, the only thing to do is to import in your contract the interface `IRuleEngine` (CMTAT) or `IERC3643Compliance` (ERC-3643), which declares the corresponding functions to call by the token contract. This interface can be found here.

Like CMTAT

Before each ERC-20 transfer, the CMTAT calls the function `transferred` which is the entrypoint for the RuleEngine.

```
function transferred(address from,address to,uint256 value)
```

If you want to apply restriction on the spender address, you have to call the `transferred` function which takes the spender argument in your ERC-20 function `transferFrom`.

```
function transferred(address spender,address from,address to,uint256 value)
```

For example, CMTAT defines the interaction with the RuleEngine inside a specific module, ValidationModuleRuleEngine and CMTATBaseRuleEngine.

- ValidationModuleRuleEngine

```
/* ============ State functions ============ */
function _transferred(address spender, address from, address to, uint256 value) internal virtual returns (bool){
    if(!_canTransferGenericByModule(spender, from, to)){
        return false;
    } else {
         IRuleEngine ruleEngine_ = ruleEngine();
         if (address(ruleEngine_) != address(0)){
             if(spender != address(0)){
                ruleEngine_.transferred(spender, from, to, value);
              } else {
                 ruleEngine_.transferred(from, to, value);
              }
        }
    }
    return true;
}
```

- CMTATBaseRuleEngine

```
function _checkTransferred(address spender, address from, address to, uint256 value) internal virtual override(CMTATBaseCommon) {
    CMTATBaseCommon._checkTransferred(spender, from, to, value);
    require(ValidationModuleRuleEngine._transferred(spender, from, to, value), Errors.CMTAT_InvalidTransfer(from, to, value));
}
```

This function `_transferred` is called before each transfer/burn/mint through the internal function `_checkTransferred` defined in CMTAT_BASE.

Like ERC-3643

The ERC-3643 defines several functions used as entrypoint for an ERC-3643 token.

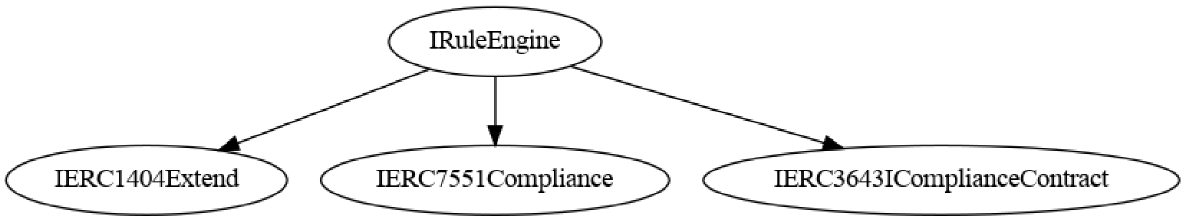
They are the following:

```
// read-only function
function canTransfer(address from, address to, uint256 value) external view
returns (bool);
// ERC-20 transfer
function transferred(address from, address to, uint256 value) external;
// mint
function created(address to, uint256 value) external;
// burn
function destroyed(address from, uint256 value) external;
```

Interface

CMTAT

The `RuleEngine` base interface is defined in CMTAT repository.



It inherits from several others interface: `IERC1404Extend`, `IERC7551Compliance`, `IERC3643ComplianceContract`

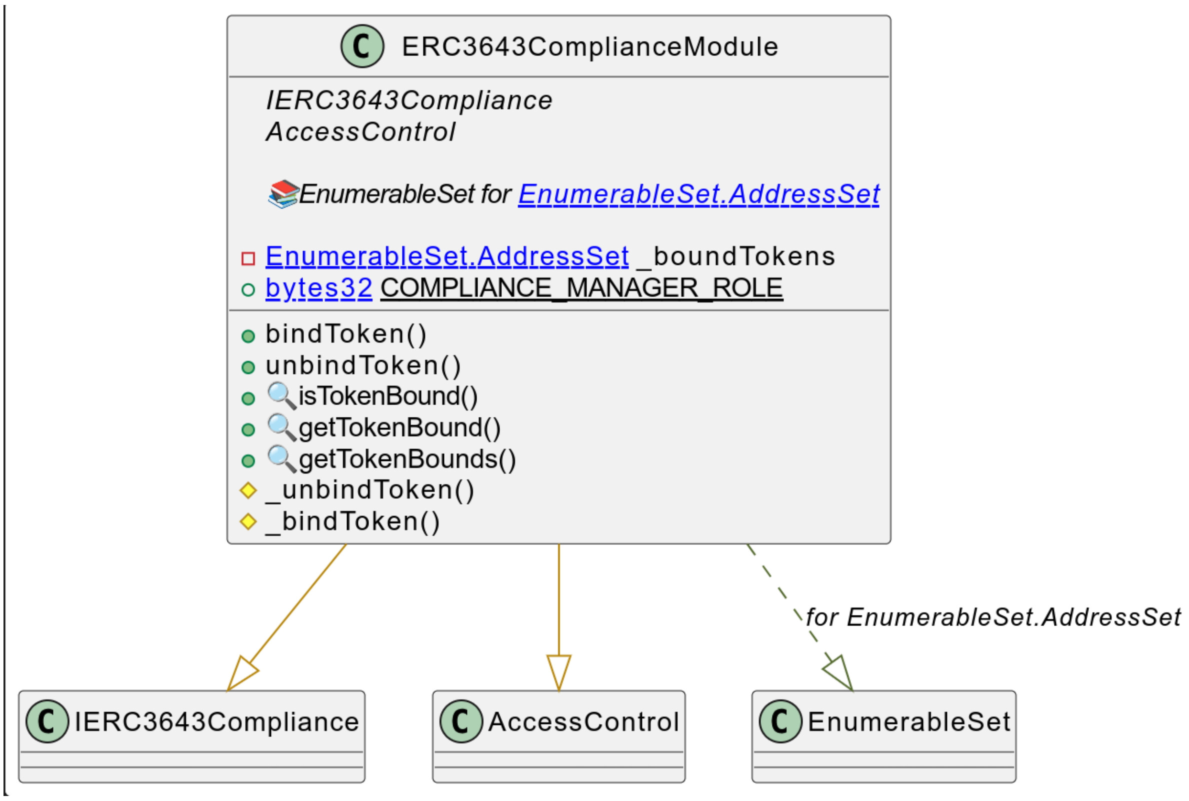
```
// IRuleEngine
```

```
function transferred(address spender, address from, address to, uint256 value)
external;

// IERC-1404
function detectTransferRestriction(address from,address to,uint256 value)
external view returns (uint8);

function messageForTransferRestriction(uint8 restrictionCode)
external view returns (string memory);

// IERC-1404Extend
enum REJECTED_CODE_BASE {
        TRANSFER_OK,
        TRANSFER_REJECTED_DEACTIVATED,
        TRANSFER_REJECTED_PAUSED,
        TRANSFER_REJECTED_FROM_FROZEN,
        TRANSFER_REJECTED_TO_FROZEN,
        TRANSFER_REJECTED_SPENDER_FROZEN,
        TRANSFER_REJECTED_FROM_INSUFFICIENT_ACTIVE_BALANCE
    }

function detectTransferRestrictionFrom(address spender,address from,address
to,uint256 value)
external view returns (uint8);


// IERC7551Compliance
function canTransferFrom(address spender,address from,address to,uint256 value)
external view returns (bool);


// IER3643ComplianceRead
function canTransfer(address from,address to,uint256 value)
external view returns (bool isValid);

// IERC3643IComplianceContract
function transferred(address from, address to, uint256 value)
external;
```

ERC-3643

The ERC-3643 compliance interface is defined in IERC3643Compliance.sol.

A specific module implements this interface for the RuleEngine: ERC3643Compliance.sol



Technical

Dependencies

The toolchain includes the following components, where the versions are the latest ones that we tested:

- Foundry (forge-std) v1.10.0
- Solidity 0.8.30 (via solc-js)
- OpenZeppelin Contracts (submodule) v5.4.0
- CMTAT v3.0.0-rc7

Contracts Description Table

Access Control (RBAC)

CMTAT uses a RBAC access control by using the contract `AccessControl` from OpenZeppelin.

Each module defines the roles useful to restrict its functions.

The `AccessControlModule` which is used by all base and deployment contracts override the OpenZeppelin function `hasRole` to give by default all the roles to the `admin`.

See also docs.openzeppelin.com - AccessControl

Role list

Here is the list of roles and their 32 bytes identifier.

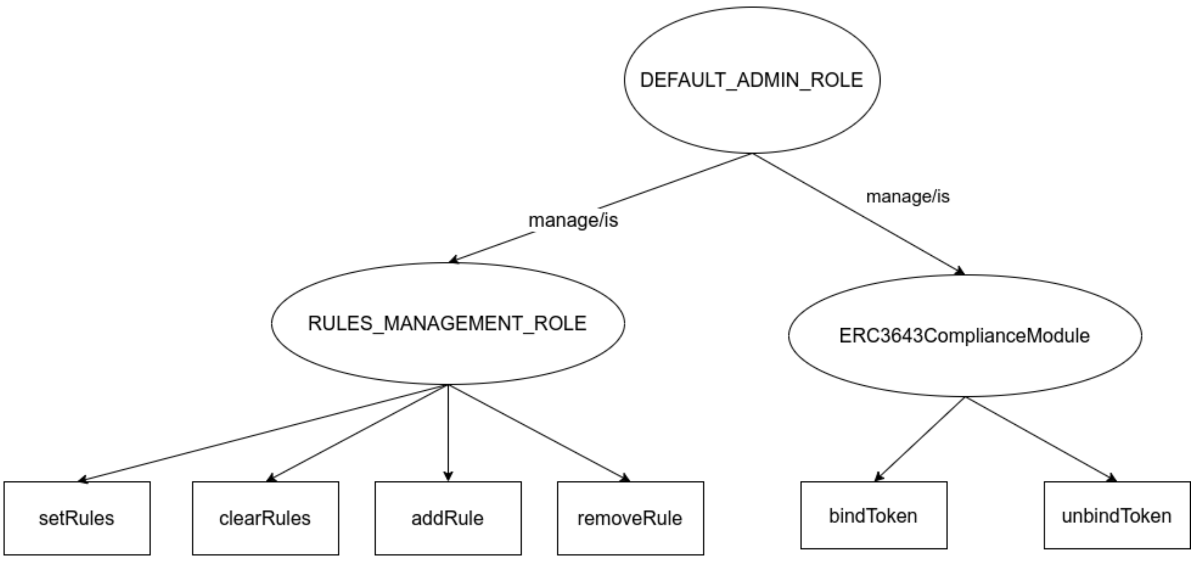
The default admin is the address put in argument(`admin`) inside the constructor.

It is set in the constructor when the contract is deployed.

	Defined in	**32 bytes identifier**
DEFAULT_ADMIN_ROLE	OpenZeppelin AccessControl	0x0000000000000000000000000000000000000000000000000000000000000000
Modules		
COMPLIANCE_MANAGER_ROLE	ERC3643Compliance	0xe5c50d0927e06141e032cb9a67e1d7092dc85c0b0825191f7e1cede600028568
RULES_MANAGEMENT_ROLE	RuleEngineInvariantStorageCommon	0xea5f4eb72290e50c32abd6c23e45de3d8300b3286e1cbc2e293114b92e034e5e

Schema

Here a schema of the Access Control.



Role by modules

Here a summary tab for each restricted functions defined in a module
For function signatures, struct arguments are represented with their corresponding native type.

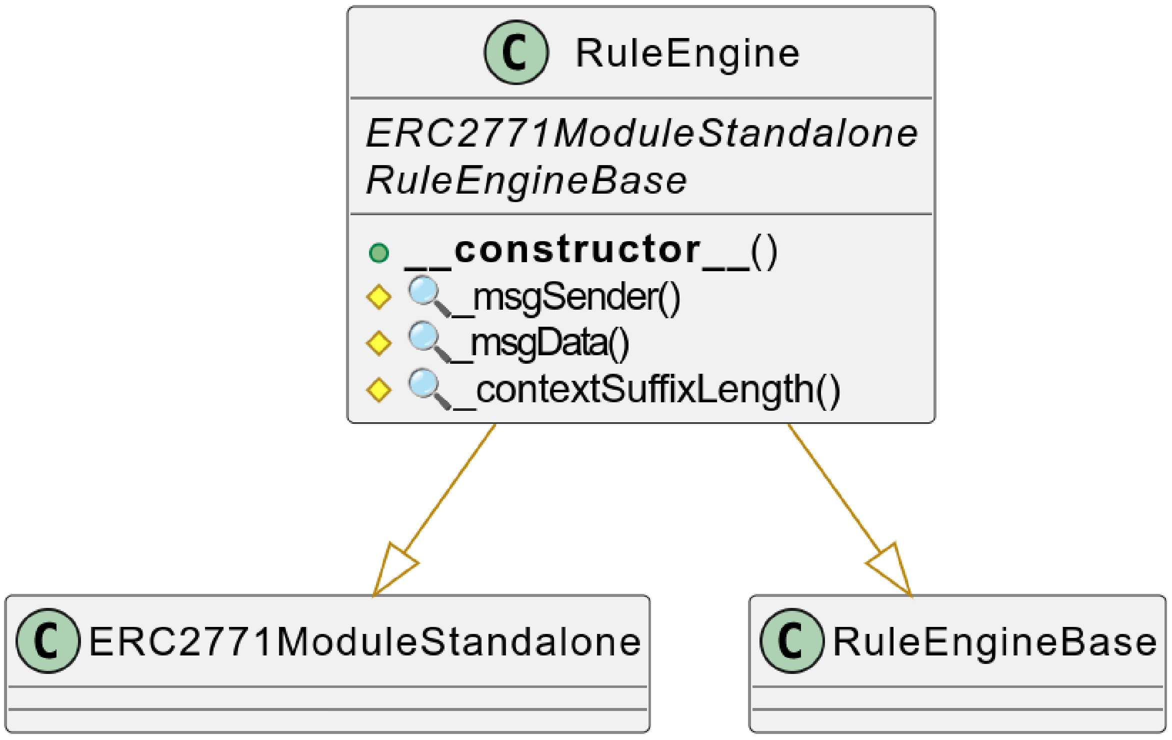
	Function signature	**Visibility [public/external]**	**Input variables (Function arguments)**	**Output variables (return value)**	**Role Required**
Modules					
RulesManagementModule					
	`setRules(address[] rules_)`	public	`IRule[] rules_`	-	RULES_MANAGEMENT_ROLE
	`clearRules()`	public	-	-	RULES_MANAGEMENT_ROLE
	`addRule(address rule_)`	public	`IRule rule_`	-	RULES_MANAGEMENT_ROLE
	`removeRule(address rule_)`	public	`IRule rule_`	-	RULES_MANAGEMENT_ROLE
ERC3643ComplianceModule					
	`bindToken(address token)`	public	`address token`	-	COMPLIANCE_MANAGER_ROLE
	`unbindToken(address token)`	public	`address token`	-	COMPLIANCE_MANAGER_ROLE
RuleEngineBase					
	`transferred(address from,address to,uint256 value)`	public	`address from,address to, uint256 value`	-	onlyBoundToken (modifier)

	Function signature	Visibility [public/external]	Input variables (Function arguments)	Output variables (return value)	Role Required
	`transferred(address spender,address from,address to,uint256 value)`	public	`address spender,address from,address to, uint256 value`	-	onlyBoundToken (modifier)

Schema

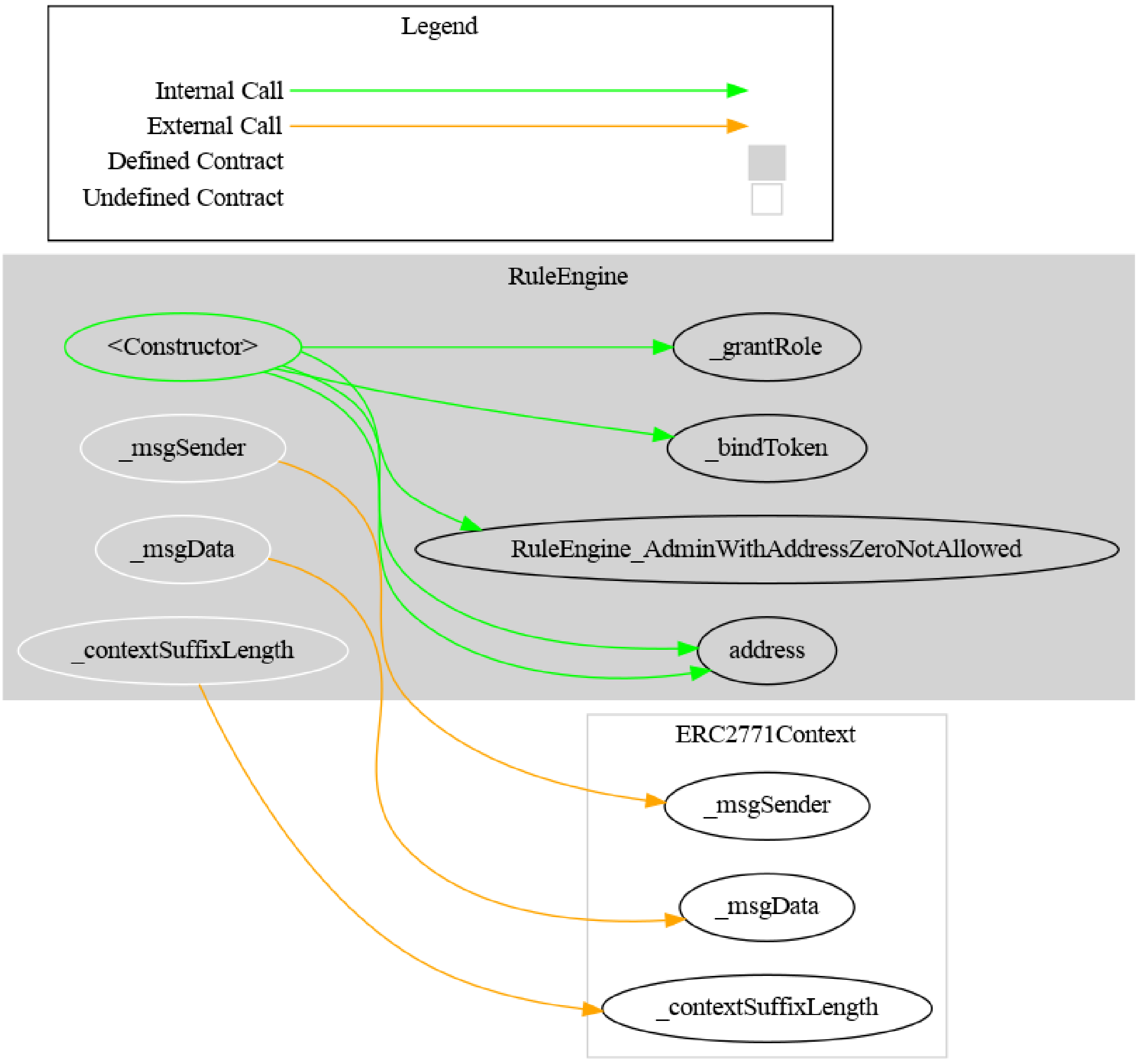
UML

Here is the UML of the main contract:



Graph

Here is the surya graphy of the main contract:



Functionality

Several functionalities are not implemented because it makes more sense to directly implement them in the token smart contract

The RuleEngine can be removed from the main token contract by calling these dedicated functions

- CMTAT v3.0.0: `setRuleEngine(address ruleEngine)`
- ERC-3643 token: `setCompliance(address _compliance)`

Available Rules

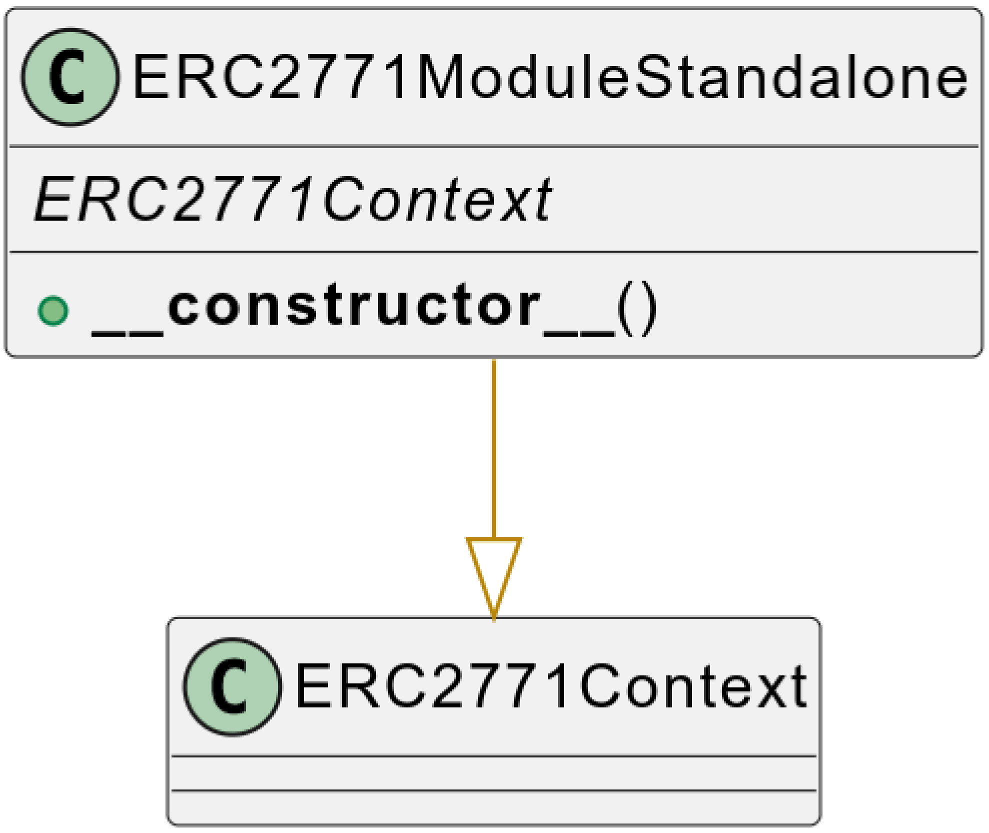
Rules have their own dedicated repository: github.com/CMTA/Rules

The following rules are available:

Rule	Type [ready-only / read-write]	Audit planned	Description

Rule	Type [ready-only / read-write]	Audit planned	Description
RuleWhitelist	Ready-only	☑	This rule can be used to restrict transfers from/to only addresses inside a whitelist.
RuleWhitelistWrapper	Ready-only	☑	This rule can be used to restrict transfers from/to only addresses inside a group of whitelist rules managed by different operators.
RuleBlacklist	Ready-only	☑	This rule can be used to forbid transfer from/to addresses in the blacklist
RuleSanctionList	Ready-only	☑	The purpose of this contract is to use the oracle contract from Chainalysis to forbid transfer from/to an address included in a sanctions designation (US, EU, or UN).
RuleConditionalTransfer	Ready-Write	☒ (experimental rule)	This rule requires that transfers have to be approved before being executed by the token holders.

Gasless support (ERC-2771)



The RuleEngine supports client-side gasless transactions using the standard ERC-2771.

The contract uses the OpenZeppelin contract `ERC2771ContextUpgradeable`, which allows a contract to get the original client with `_msgSender()` instead of the feepayer given by `msg.sender`.

At deployment, the parameter `forwarder` inside the RuleEngine contract constructor has to be set with the defined address of the forwarder.

After deployment, the forwarder is immutable and can not be changed.

References:

- OpenZeppelin Meta Transactions
- OpenGSN has deployed several forwarders, see their documentation to see some examples.

Upgradeable

A proxy architecture (upgradeable) increases the code complexity as well as the runtime gas cost for each transaction. This is why the RuleEngine is not upgradeable.

Moreover, in a proxy architecture, each new implementation must be compatible (storage) with the precedent implementation, which can reduce the ability to improve the code.

In case you use the same RuleEngine for several different tokens, unfortunately, you will have to update the address of the RuleEngine set in each token contract separately.

Urgency mechanism

Pause

There are no functionalities to put in pause the RuleEngine.

The RuleEngine can be removed from the main token contract by calling the dedicated functions to manage the RuleEngine

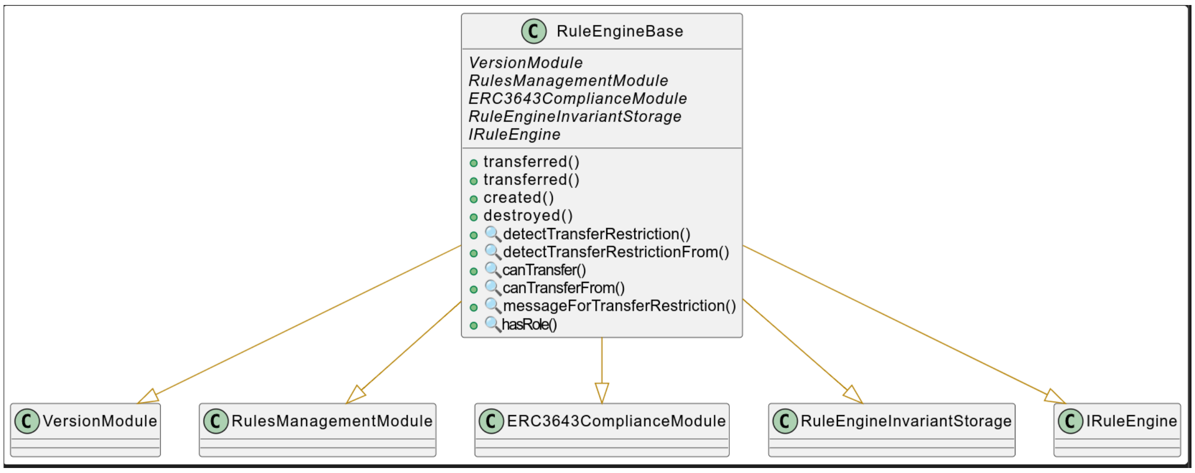
Kill / Deactivate the contracts

There are no functionalities to kill/deactivate the contracts.

Similar to the pause functionality, the RuleEngine can be directly removed from the main token contract.

Ethereum API

RuleEngineBase



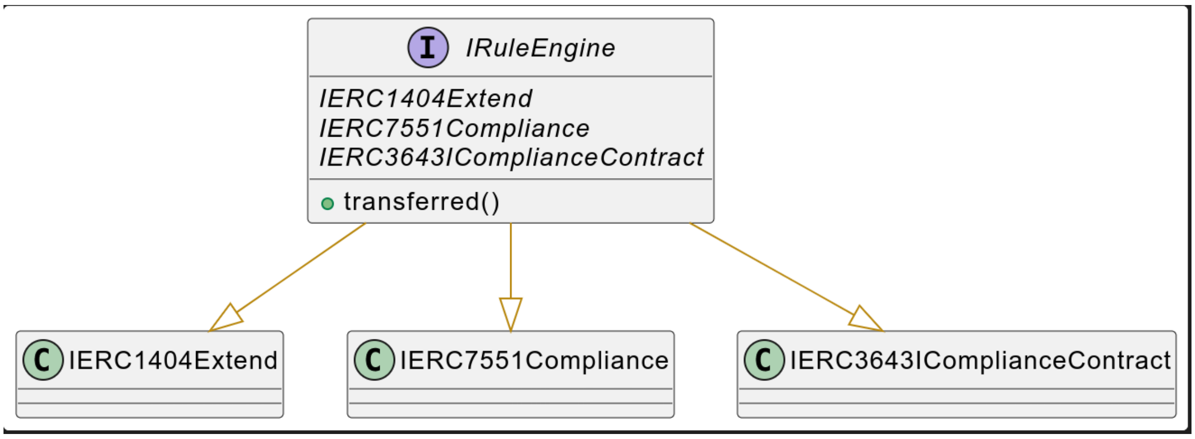
Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
RuleEngineBase	Implementation	VersionModule, RulesManagementModule, ERC3643ComplianceModule, RuleEngineInvariantStorage, IRuleEngine		
└	transferred	Public ❗️	🛑	onlyBoundToken
└	transferred	Public ❗️	🛑	onlyBoundToken
└	created	Public ❗️	🛑	onlyBoundToken
└	destroyed	Public ❗️	🛑	onlyBoundToken
└	detectTransferRestriction	Public ❗️		NO ❗️
└	detectTransferRestrictionFrom	Public ❗️		NO ❗️
└	canTransfer	Public ❗️		NO ❗️
└	canTransferFrom	Public ❗️		NO ❗️
└	messageForTransferRestriction	Public ❗️		NO ❗️
└	hasRole	Public ❗️		NO ❗️

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

IRuleEngine



transferred(address spender, address from, address to, uint256 value)

```
function transferred(address spender,address from,address to,uint256 value)
public virtual override(IRuleEngine)
onlyBoundToken
```

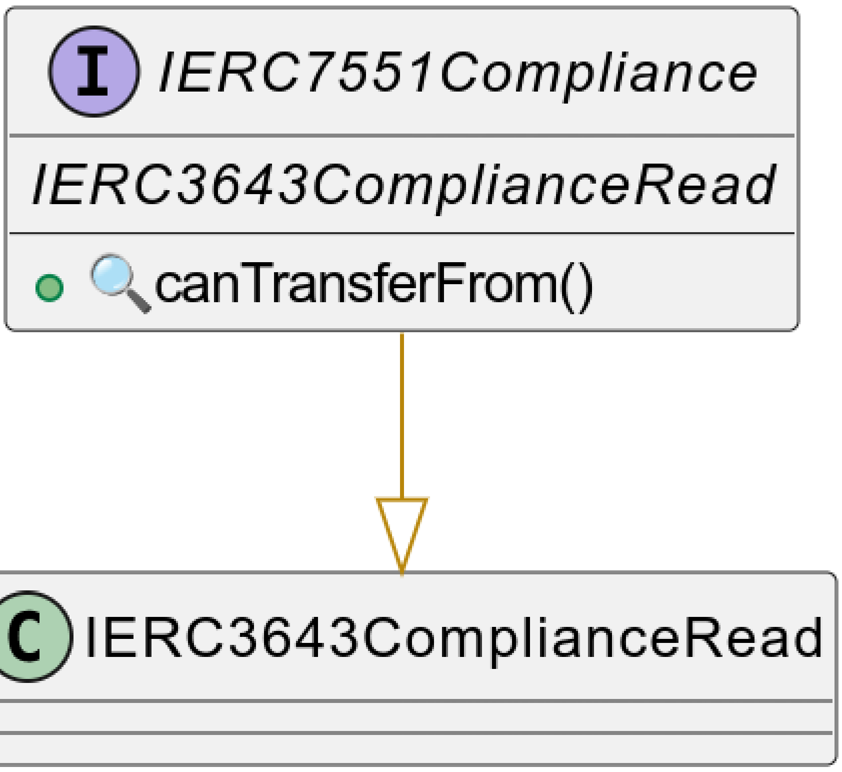
Function called whenever tokens are transferred from one wallet to another.

Must revert if the transfer is invalid.
Same name as ERC-3643 but with an additional `spender` parameter.
This function can be used to update state variables of the RuleEngine contract.
Can only be called by the token contract bound to the RuleEngine.

Input Parameters:

Name	Type	Description
spender	address	The spender address initiating the transfer.
from	address	The token holder address.
to	address	The receiver address.
value	uint256	The amount of tokens involved in the transfer.

IERC7551Compliance



canTransferFrom(address spender, address from, address to, uint256 value) -> bool

Checks if `spender` can transfer `value` tokens from `from` to `to` under compliance rules.

Does not check balances or access rights (Access Control).

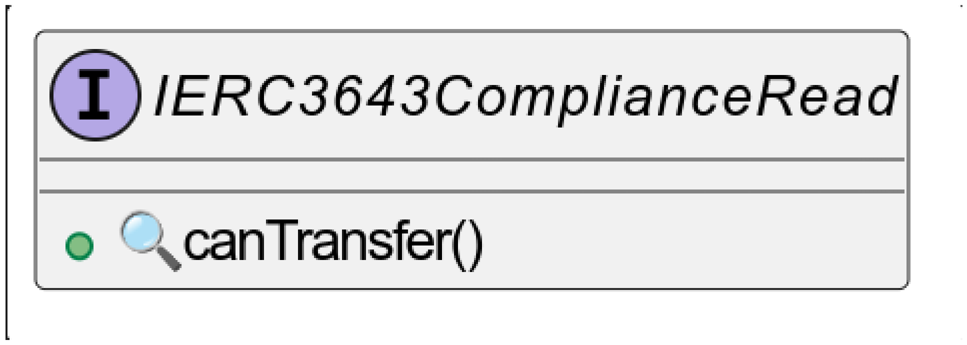
Input Parameters:

Name	Type	Description
spender	address	The address performing the transfer.
from	address	The source address.
to	address	The destination address.
value	uint256	The number of tokens to transfer.

Return Values:

Type	Description
bool	True if the transfer complies with policy.

IERC3643ComplianceRead



canTransfer(address from, address to, uint256 value) -> bool

Returns true if the transfer is valid, and false otherwise.

Does not check balances or access rights (Access Control).

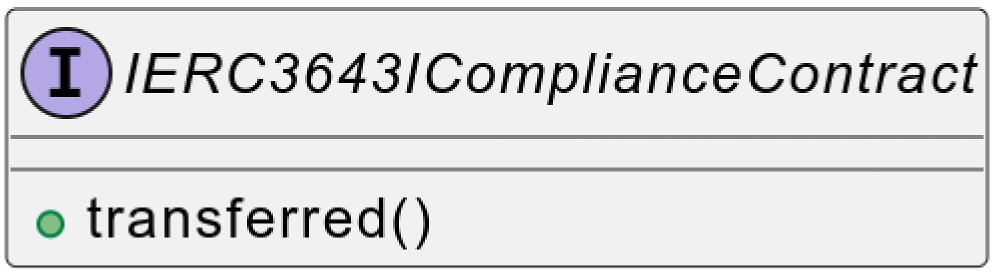
Input Parameters:

Name	Type	Description
from	address	The source address.
to	address	The destination address.
value	uint256	The number of tokens to transfer.

Return Values:

Type	Description
bool	True if the transfer is valid, false otherwise.

IERC3643IComplianceContract



transferred(address from, address to, uint256 value)

```
function transferred(address from,address to,uint256 value)
public virtual override(IERC3643IComplianceContract)
onlyBoundToken
```

Updates the compliance contract state whenever tokens are transferred.

Can only be called by the token contract bound to this compliance logic.
This function can be used to update internal state variables.

Input Parameters:

Name	Type	Description
from	address	The address of the sender.
to	address	The address of the receiver.
value	uint256	The number of tokens involved in the transfer.

IERC3643Compliance

created(address to, uint256 value)

```
function created(address to, uint256 value)
public virtual override(IERC3643Compliance)
onlyBoundToken
```

Updates the compliance contract state when tokens are created (minted).

Called by the token contract when new tokens are issued to an account.
Reverts if the minting does not comply with the rules.

Input Parameters:

Name	Type	Description
to	address	The address receiving the minted tokens.
value	uint256	The number of tokens created.

destroyed(address from, uint256 value)

```
function destroyed(address from, uint256 value)
public virtual override(IERC3643Compliance)
onlyBoundToken
```

Updates the compliance contract state when tokens are destroyed (burned).

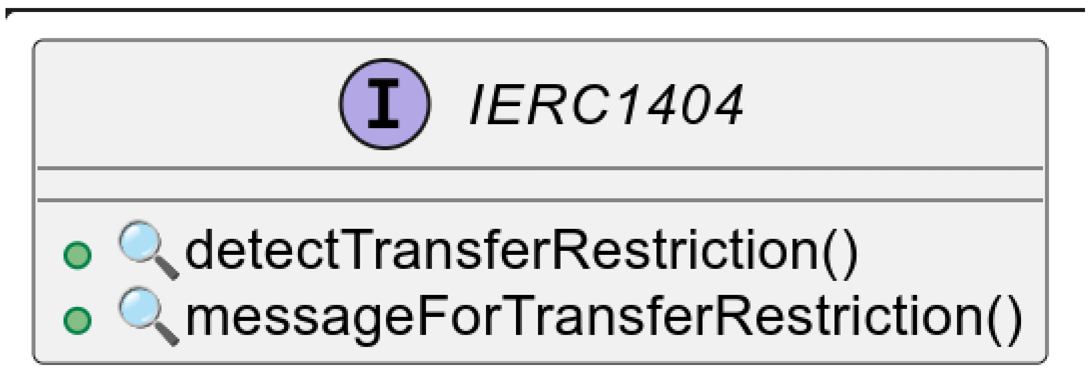
Called by the token contract when tokens are redeemed or burned.
Reverts if the burning does not comply with the rules.

Input Parameters:

Name	Type	Description

Name	Type	Description
from	address	The address whose tokens are being destroyed.
value	uint256	The number of tokens destroyed.

IERC1404



detectTransferRestriction(address from, address to, uint256 value) -> uint8

Returns a uint8 code to indicate if a transfer is restricted or not.

Implements the restriction logic of {ERC-1404}.
 Examples of restriction logic include:

- checking if the recipient is whitelisted,
- checking if the sender’s tokens are frozen during a lock-up period, etc.

Input Parameters:

Name	Type	Description
from	address	The source address.
to	address	The destination address.
value	uint256	The number of tokens to transfer.

Return Values:

Type	Description
uint8	Restriction code (0 means the transfer is authorized).

messageForTransferRestriction(uint8 restrictionCode) -> string

Returns a human-readable explanation for a transfer restriction code.

Implements {ERC-1404} standard message accessor.

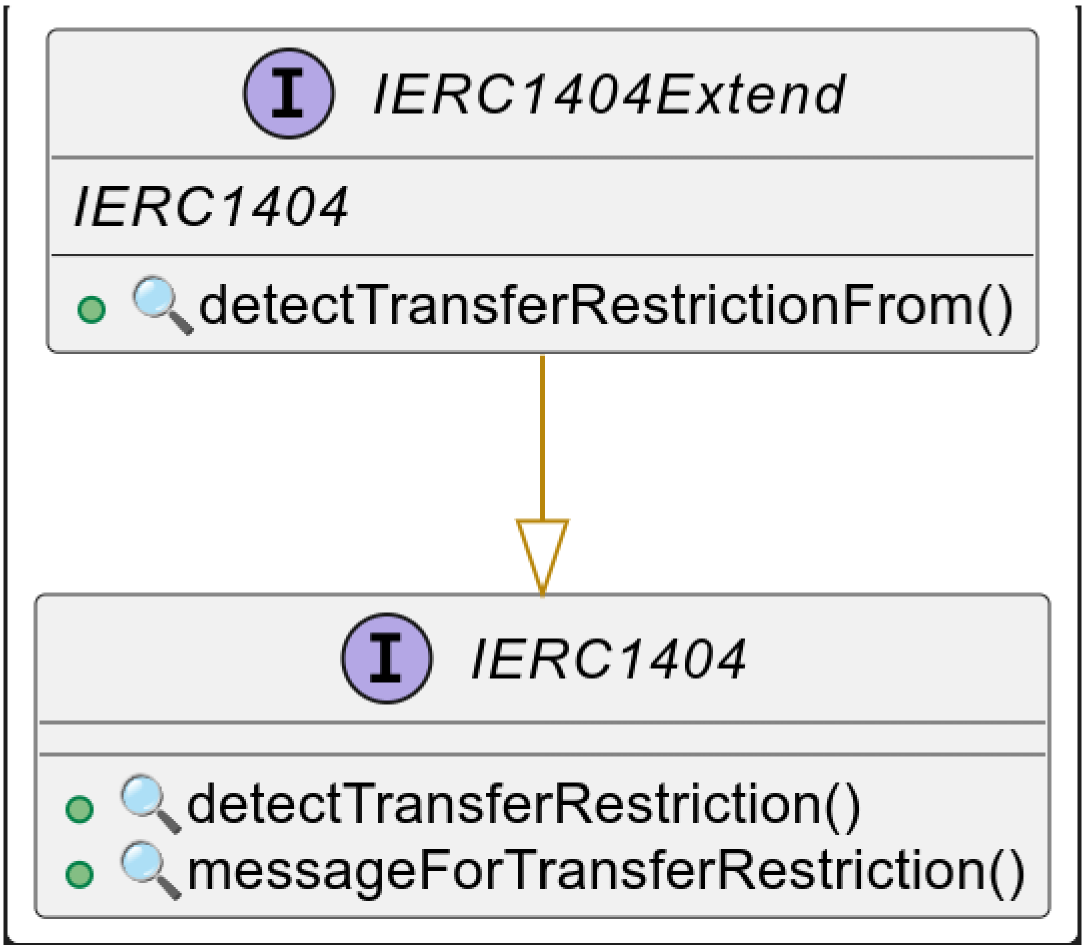
Input Parameters:

Name	Type	Description
restrictionCode	uint8	The restriction code to interpret.

Return Values:

Type	Description
string	A message describing why the transfer is restricted.

IERC1404Extend


IERC1404Extend
IERC1404
detectTransferRestrictionFrom()
IERC1404
detectTransferRestriction()
messageForTransferRestriction()

enum REJECTED_CODE_BASE

Error codes for transfer restrictions.
Codes `6-9` are reserved for future CMTAT ruleEngine extensions.

Name	Value	Description
TRANSFER_OK	0	Transfer authorized.
TRANSFER_REJECTED_PAUSED	1	Transfer rejected because the token is paused.
TRANSFER_REJECTED_FROM_FROZEN	2	Transfer rejected because the sender's address is frozen.
TRANSFER_REJECTED_TO_FROZEN	3	Transfer rejected because the recipient's address is frozen.
TRANSFER_REJECTED_SPENDER_FROZEN	4	Transfer rejected because the spender's address is frozen.
TRANSFER_REJECTED_FROM_INSUFFICIENT_ACTIVE_BALANCE	5	Transfer rejected because the sender does not have enough active (unfrozen) balance.

---

detectTransferRestrictionFrom(address spender, address from, address to, uint256 value) -> uint8

Returns a uint8 code to indicate if a transfer is restricted or not.

This is an extension of {ERC-1404} with an additional `spender` parameter to enforce restriction logic on delegated transfers.
Examples of restriction logic include:

- verifying if the recipient is whitelisted,
- verifying if tokens are locked for either sender or spender, etc.

Input Parameters:

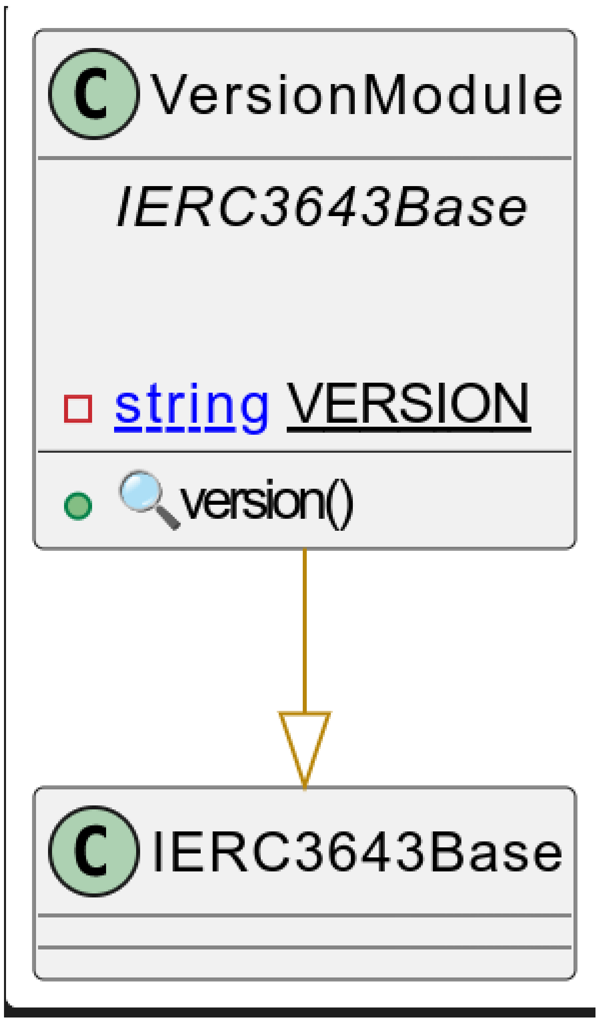
Name	Type	Description
spender	address	The address initiating the transfer (for delegated transfers).

Name	Type	Description
from	address	The source address.
to	address	The destination address.
value	uint256	The number of tokens to transfer.

Return Values:

Type	Description
uint8	Restriction code (0 means the transfer is authorized).

VersionModule



Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**

Contract	Type	Bases		
VersionModule	Implementation	IERC3643Base		
L	version	Public ❗		NO ❗

version()

```
function version() external view returns (string memory version_);
```

```
function version()
public view virtual override(IERC3643Base)
returns (string memory version_)
```

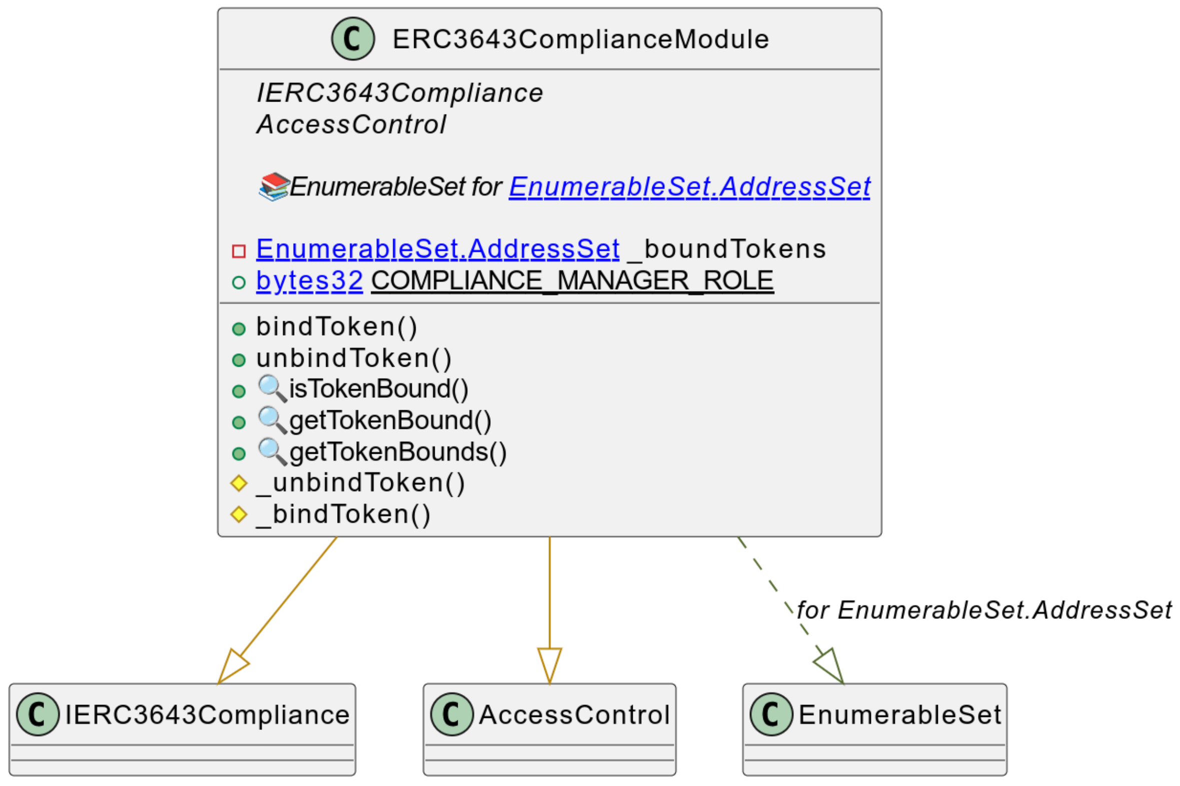
Description

Returns the current version of the token contract.
Useful for identifying which version of the smart contract is deployed and in use.

Return

Name	Type	Description
`version_`	string	The version string of the token implementation (e.g., "1.0.0").

ERC3643ComplianceModule



Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
ERC3643ComplianceModule	Implementation	IERC3643Compliance, AccessControl		
└	bindToken	Public ❗️	🛑	onlyRole
└	unbindToken	Public ❗️	🛑	onlyRole
└	isTokenBound	Public ❗️		NO❗️
└	getTokenBound	External ❗️		NO❗️
└	getTokenBounds	External ❗️		NO❗️
└	_unbindToken	Internal 🔒	🛑	
└	_bindToken	Internal 🔒	🛑	

Events

TokenBound(address token)

```
event TokenBound(address token)
```

Emitted when a token is successfully bound to the compliance contract.

Event Parameters:

Name	Type	Description
token	address	The address of the token that was bound.

TokenUnbound(address token)

```
event TokenUnbound(address token)
```

Emitted when a token is successfully unbound from the compliance contract.

Event Parameters:

Name	Type	Description
token	address	The address of the token that was unbound.

Functions

bindToken(address token)

```
function bindToken(address token)
public override virtual
onlyRole(COMPLIANCE_MANAGER_ROLE)
```

Associates a token contract with this compliance contract.

The compliance contract may restrict operations on the bound token according to its internal compliance logic.
 Reverts if the token is already bound.

Input Parameters:

Name	Type	Description
token	address	The address of the token to bind.

---

unbindToken(address token)

```
function unbindToken(address token)
public override virtual
onlyRole(COMPLIANCE_MANAGER_ROLE)
```

Removes the association of a token contract from this compliance contract.

Reverts if the token is not currently bound.

Input Parameters:

Name	Type	Description
token	address	The address of the token to unbind.

---

isTokenBound(address token) -> bool

```
function isTokenBound(address token)
public view virtual override
returns (bool)
```

Checks whether a token is currently bound to this compliance contract.

Input Parameters:

Name	Type	Description

Name	Type	Description
token	address	The token address to verify.

Return Values:

Type	Description
bool	True if the token is bound, false otherwise.

---

getTokenBound() -> address

```
function getTokenBound()
public view virtual override
returns (address)
```

Returns the single token currently bound to this compliance contract.

If multiple tokens are supported, consider using `getTokenBounds()`.

Note that there are no guarantees on the ordering of values inside the array, and it may change when more values are added or removed.

Return Values:

Type	Description
address	The address of the currently bound token.

---

getTokenBounds() -> address[]

```
function getTokenBounds()
public view override
returns (address[] memory)
```

Returns all tokens currently bound to this compliance contract.

This is a view-only function and does not modify state.
This function is not part of the original ERC-3643 specification.

This operation will copy the entire storage to memory, which can be quite expensive.

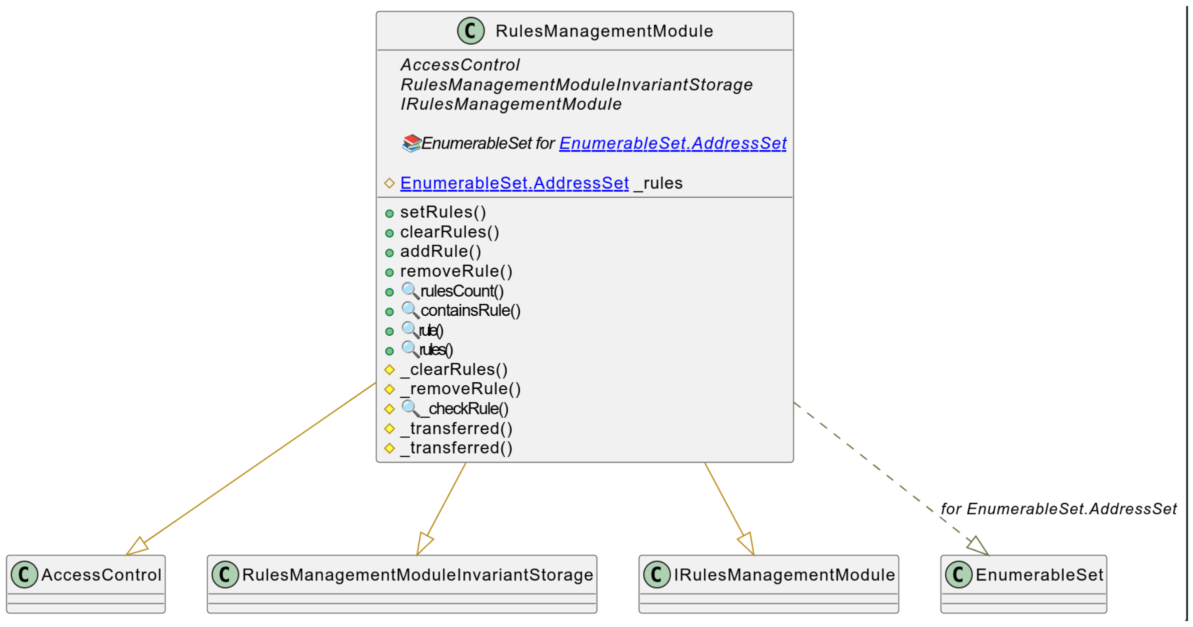
This is designed to mostly be used by view accessors that are queried without any gas fees.

Return Values:

Type	Description

Type	Description
address[]	An array of addresses of bound token contracts.

RulesManagementModule



Contracts Description Table

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
RulesManagementModule	Implementation	AccessControl, RulesManagementModuleInvariantStorage, IRulesManagementModule		
L	setRules	Public ❗	🛑	onlyRole
L	clearRules	Public ❗	🛑	onlyRole
L	addRule	Public ❗	🛑	onlyRole
L	removeRule	Public ❗	🛑	onlyRole
L	rulesCount	Public ❗		NO ❗
L	containsRule	Public ❗		NO ❗
L	rule	Public ❗		NO ❗
L	rules	Public ❗		NO ❗
L	_clearRules	Internal 🔒	🛑	
L	_removeRule	Internal 🔒	🛑	
L	_checkRule	Internal 🔒		
L	_transferred	Internal 🔒	🛑	
L	_transferred	Internal 🔒	🛑	

Events

event AddRule(address rule)

```
event AddRule(IRule indexed rule)
```

Emitted when a new rule is added to the rule set.

Event Parameters:

Name	Type	Description
rule	IRule	The address of the rule contract that was added.

---

event RemoveRule(address rule)

```
event RemoveRule(IRule indexed rule)
```

Emitted when a rule is removed from the rule set.

Event Parameters:

Name	Type	Description
rule	IRule	The address of the rule contract that was removed.

---

event ClearRules()

```
event ClearRules()
```

Emitted when all rules are cleared from the rule set.

This event has no parameters.

Functions

setRules(address[] rules_)

```
function setRules(IRule[] calldata rules_)
public virtual override(IRulesManagementModule)
onlyRole(RULES_MANAGEMENT_ROLE)
```

Defines the complete list of rules for the rule engine.

Any previously configured rules are completely replaced.
Rules must be deployed contracts implementing the expected `IRule` interface.
Reverts if any rule address is zero or if duplicates are detected.

This function calls _clearRules if at least one rule is still configured

Input Parameters:

Name	Type	Description
rules_	IRule[]	The array of IRule contracts to configure as the active rules.

---

rulesCount() -> uint256

```
function rulesCount()
public view virtual override(IRulesManagementModule)
returns (uint256)
```

Returns the total number of currently configured rules.

Equivalent to the length of the internal rules array.

Return Values:

Type	Description
uint256	The number of active rules.

---

rule(uint256 ruleId) -> address

```
function rule(uint256 ruleId)
public view virtual override(IRulesManagementModule)
returns (address)
```

Retrieves the rule address at a specific index.

Return the `zero address` is out of bounds.

Note that there are no guarantees on the ordering of values inside the array, and it may change when more values are added or removed.

Input Parameters:

Name	Type	Description
ruleId	uint256	The index of the desired rule in the array.

Return Values:

Type	Description
address	The address of the corresponding IRule contract.

---

rules() -> address[]

```
function rules()
public view virtual override(IRulesManagementModule)
returns (address[] memory)
```

Returns the full list of currently configured rules.

This is a view-only function and does not modify state.

This operation will copy the entire storage to memory, which can be quite expensive.

This is designed to mostly be used by view accessors that are queried without any gas fees.

Return Values:

Type	Description
address[]	An array containing all active rule contract addresses.

---

clearRules()

```
function clearRules()
public virtual override(IRulesManagementModule)
onlyRole(RULES_MANAGEMENT_ROLE)
```

Removes all configured rules.

After calling this function, no rules will remain set.

Developers should keep in mind that this function has an unbounded cost and using it may render the function uncallable if the set grows to the point where clearing it consumes too much gas to fit in a block.

---

addRule(address rule_)

```
function addRule(IRule rule_)
public virtual override(IRulesManagementModule)
onlyRole(RULES_MANAGEMENT_ROLE)
```

Adds a new rule to the current rule set.

Reverts if the rule address is zero or already exists in the set.

Input Parameters:

Name	Type	Description
rule_	IRule	The IRule contract to add.

---

removeRule(address rule_)

```
function removeRule(IRule rule_)
public virtual
override(IRulesManagementModule)
onlyRole(RULES_MANAGEMENT_ROLE)
```

Removes a specific rule from the current rule set.

Reverts if the provided rule is not found or does not match the stored rule at its index.

Input Parameters:

Name	Type	Description
rule_	IRule	The IRule contract to remove.

---

containsRule(address rule_) -> bool

```
function containsRule(IRule rule_)
public view virtual override(IRulesManagementModule)
returns (bool)
```

Checks whether a specific rule is currently configured.

Input Parameters:

Name	Type	Description
rule_	IRule	The IRule contract to check for membership.

Return Values:

Type	Description
bool	True if the rule is present, false otherwise.

Security

Vulnerability disclosure

Please see SECURITY.md (CMTAT main repository).

The contracts have been audited by ABDKConsulting, a globally recognized firm specialized in smart contracts' security.

Audit

First Audit - March 2022

Fixed version : v1.0.2

The first audit was performed by ABDK on the version 1.0.1.

The release v1.0.2 contains the different fixes and improvements related to this audit.

The temporary report is available in Taurus. Audit 3.3.CollectedIssues.ods

The final report is available in ABDK_CMTA_CMTATRuleEngine_v_1_0.pdf.

Tools

Slither

Here is the list of report performed with Slither

Version	File
latest	slither-report.md

```
slither .  --checklist --filter-paths "openzeppelin-contracts|test|CMTAT|forge-std|mocks" > slither-report.md
```

Aderyn

Here is the list of report performed with Aderyn

```
aderyn -x mocks --output aderyn-report.md
```

Version	File
latest	aderyn-report.md

Documentation

Here a summary of the main documentation

Document	Link/Files
Toolchain	doc/TOOLCHAIN.md
Surya report	doc/schema/surya

See also Taurus - Token Transfer Management: How to Apply Restrictions with CMTAT and ERC-1404 (RuleEngine v2.02 and CMTAT v2.4.0)

Toolchains and Usage

Explain how it works.

Configuration

Here are the settings for Hardhat and Foundry.

- `hardhat.config.js`
  - Solidity v0.8.30
  - EVM version: Prague (Pectra upgrade)
  - Optimizer: true, 200 runs
- `foundry.toml`
  - Solidity v0.8.30
  - EVM version: Prague (Pectra upgrade)
  - Optimizer: true, 200 runs

Toolchain installation

The contracts are developed and tested with Foundry, a smart contract development toolchain.

To install the Foundry suite, please refer to the official instructions in the Foundry book.

Initialization

You must first initialize the submodules, with

```
forge install
```

See also the command's documentation.

Later you can update all the submodules with:

```
forge update
```

See also the command's documentation.

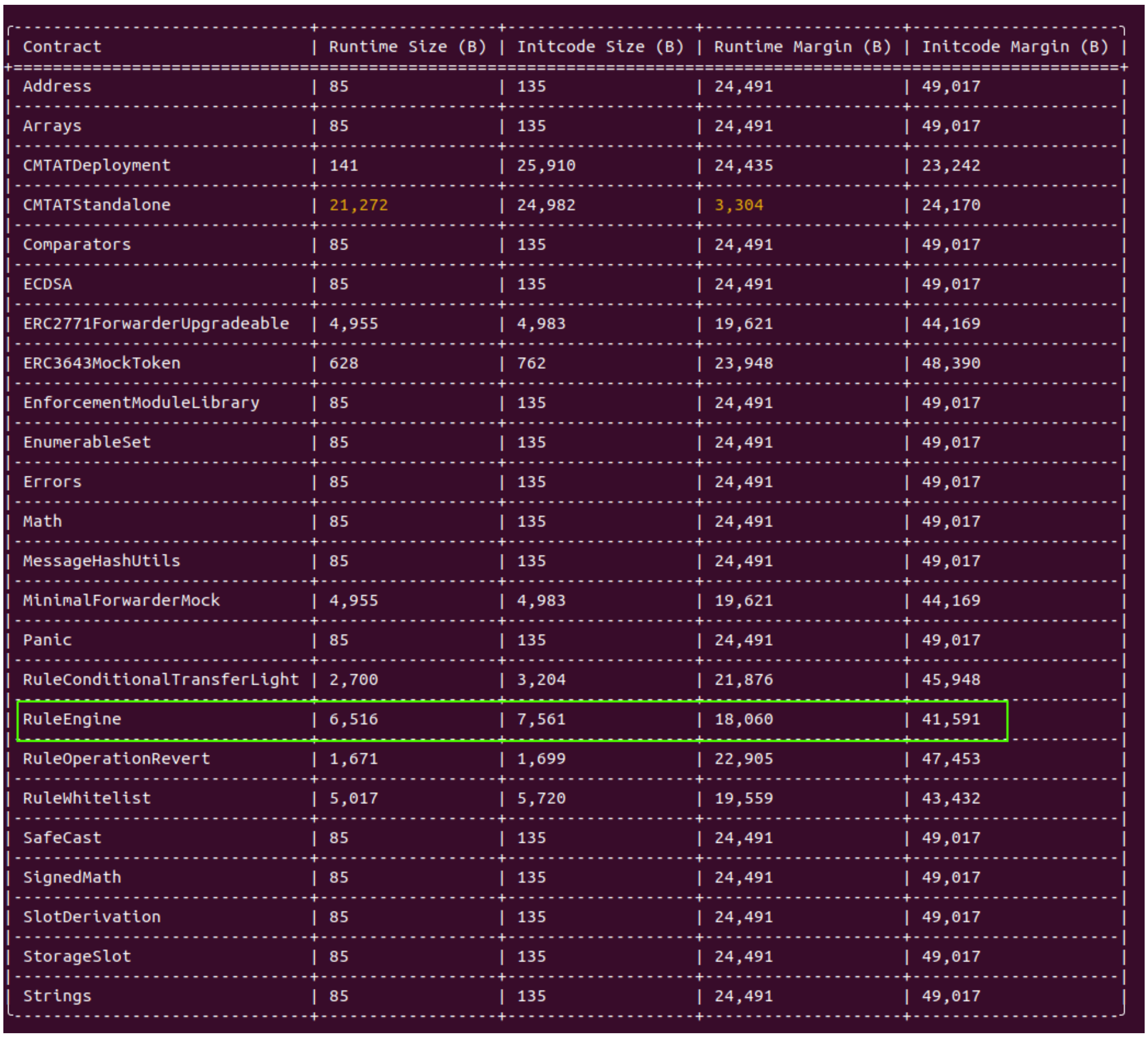
Compilation

The official documentation is available in the Foundry website

```
 forge build --contracts src/RuleEngine.sol
```

Contract size

```
 forge compile --sizes
```



```
+-------------------------------+------------------+-------------------+--------------------+---------------------+
| Contract                      | Runtime Size (B) | Initcode Size (B) | Runtime Margin (B) | Initcode Margin (B) |
+===============================================================================================================+
| Address                       | 85               | 135               | 24,491             | 49,017              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| Arrays                        | 85               | 135               | 24,491             | 49,017              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| CMTATDeployment               | 141              | 25,910            | 24,435             | 23,242              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| CMTATStandalone               | 21,272           | 24,982            | 3,304              | 24,170              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| Comparators                   | 85               | 135               | 24,491             | 49,017              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| ECDSA                         | 85               | 135               | 24,491             | 49,017              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| ERC2771ForwarderUpgradeable   | 4,955            | 4,983             | 19,621             | 44,169              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| ERC3643MockToken              | 628              | 762               | 23,948             | 48,390              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| EnforcementModuleLibrary      | 85               | 135               | 24,491             | 49,017              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| EnumerableSet                 | 85               | 135               | 24,491             | 49,017              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| Errors                        | 85               | 135               | 24,491             | 49,017              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| Math                          | 85               | 135               | 24,491             | 49,017              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| MessageHashUtils              | 85               | 135               | 24,491             | 49,017              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| MinimalForwarderMock          | 4,955            | 4,983             | 19,621             | 44,169              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| Panic                         | 85               | 135               | 24,491             | 49,017              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| RuleConditionalTransferLight  | 2,700            | 3,204             | 21,876             | 45,948              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| RuleEngine                    | 6,516            | 7,561             | 18,060             | 41,591              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| RuleOperationRevert           | 1,671            | 1,699             | 22,905             | 47,453              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| RuleWhitelist                 | 5,017            | 5,720             | 19,559             | 43,432              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| SafeCast                      | 85               | 135               | 24,491             | 49,017              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| SignedMath                    | 85               | 135               | 24,491             | 49,017              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| SlotDerivation                | 85               | 135               | 24,491             | 49,017              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| StorageSlot                   | 85               | 135               | 24,491             | 49,017              |
|-------------------------------+------------------+-------------------+--------------------+---------------------|
| Strings                       | 85               | 135               | 24,491             | 49,017              |
+-------------------------------+------------------+-------------------+--------------------+---------------------+
```

Testing

You can run the tests with

```
forge test
```

To run a specific test, use

```
forge test --match-contract <contract name> --match-test <function name>
```

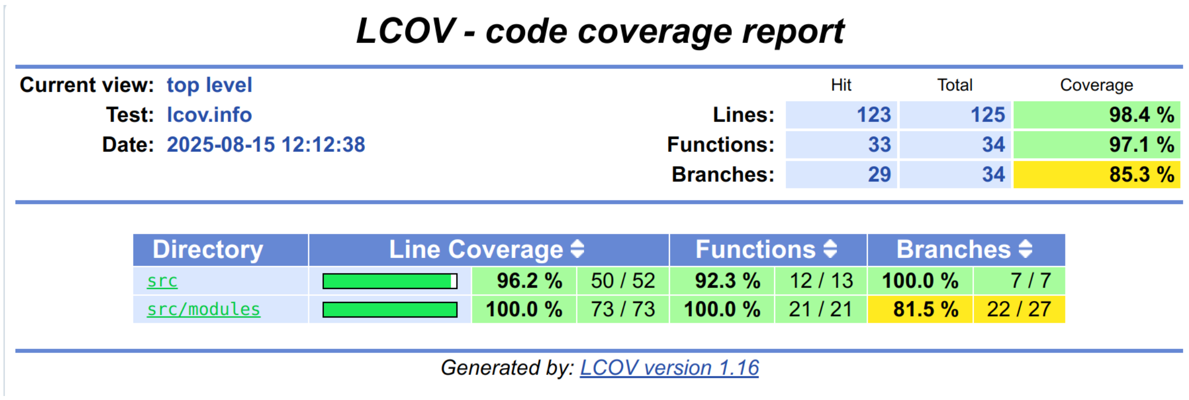
Generate gas report

```
forge test --gas-report
```

See also the test framework's official documentation, and that of the test commands.

Coverage

A code coverage is available in index.html.



LCOV - code coverage report

			Hit	Total	Coverage
Current view:	top level				
Test:	lcov.info	**Lines:**	123	125	98.4 %
Date:	2025-08-15 12:12:38	**Functions:**	33	34	97.1 %
		Branches:	29	34	85.3 %

Directory	Line Coverage			Functions		Branches	
src	🟩	96.2 %	50 / 52	92.3 %	12 / 13	100.0 %	7 / 7
src/modules	🟩	100.0 %	73 / 73	100.0 %	21 / 21	81.5 %	22 / 27

Generated by: LCOV version 1.16

- Perform a code coverage

```
forge coverage
```

- Generate LCOV report

```
forge coverage --report lcov
```

- Generate `index.html`

```
forge coverage --no-match-coverage "(script|mocks|test)" --report lcov &&
genhtml lcov.info --branch-coverage --output-dir coverage
```

See Solidity Coverage in VS Code with Foundry & Foundry forge coverage

Deployment

The official documentation is available in the Foundry website

Script

> This documentation has been written for the version v1.0.2

To run the script for deployment, you need to create a .env file. The value for CMTAT.ADDRESS is require only to use the script **RuleEngine.s.sol**
Warning : put your private key in a .env file is not the best secure way.

- File .env

```
PRIVATE_KEY=<YOUR_PRIVATE_KEY>
CMTAT_ADDRESS=<CMTAT ADDDRESS
```

- Command

CMTAT with RuleEngine

```
forge script script/CMTATWithRuleEngineScript.s.sol:CMTATWithRuleEngineScript -
-rpc-url=$RPC_URL  --broadcast --verify -vvv
```

- Value of YOUR_RPC_URL with a local instance of anvil : 127.0.0.1:8545

```
forge script script/CMTATWithRuleEngineScript.s.sol:CMTATWithRuleEngineScript --rpc-url=127.0.0.1:8545  --broadcast --verify -vvv
```

Only RuleEngine with a Whitelist contract

```
forge script script/RuleEngineScript.s.sol:RuleEngineScript --rpc-url=$RPC_URL  --broadcast --verify -vvv
```

- With anvil

```
forge script script/RuleEngineScript.s.sol:RuleEngineScript --rpc-url=127.0.0.1:8545  --broadcast --verify -vvv
```

Solidity style guideline

RuleEngine follows the solidity style guideline and the natspec format for comments

- Orders of Functions

Functions are grouped according to their visibility and ordered:

```
1. constructor

2. receive function (if exists)

3. fallback function (if exists)

4. external

5. public

6. internal

7. private
```

Within a grouping, place the `view` and `pure` functions last

- Function declaration

```
1. Visibility
2. Mutability
3. Virtual
4. Override
5. Custom modifiers
```

Intellectual property