TECHNICAL EQUIVALENCE PAPER

CMTAT and ERC-7551

30th September 2025

Summary

The CMTAT framework and Solidity reference implementation (v3.0.0) have been updated to align with the requirements of the German Electronic Securities Act (eWpG**).** This article presents the details of the changes, including a correspondence table between the eWpG compatible ERC-7551 and the equivalent functions inside the CMTAT and the Solidity implementation.

1. INTRODUCTION

In September 2025 the Capital Markets and Technology Association (CMTA) and the German Association for Electronic Securities (Bundesverband für elektronische Wertpapiere e.V.) (BfeW) announced a collaboration. A key outcome of this collaboration is the update of CMTA's smart contract framework for tokenization, the **CMTAT**, to reflect the requirements of Germany's eWpG, through alignment with the **ERC-7551** standard, developed by BfeW.

The ERC-7551 is currently an Ethereum framework proposed by the BfeW to tokenize assets in compliance with eWpG. The interface is designed to work on top of existing standards that cover the actual storage of ownership of shares of a security in the form of a token (e.g. ERC-20 or ERC-1155).

The CMTA Standard Token for Securities (CMTAT) is an open standard for smart contracts designed specifically for the tokenization of financial instruments, published by the CMTA. It is a framework that defines necessary and optional functions that can be used for tokenizing financial instruments such as equity, debt and structured products or for issuing stablecoins and is used by multiple institutional actors, including global banks. A Solidity reference implementation is available, which is an extension of the well-known ERC-20 standard. An introduction to the CMTAT is available on the CMTA website and the Solidity reference implementation is available as open-source code on GitHub.

2. STREAMLINING STANDARDS

To fully realize the benefits of tokenization, it is important to establish unified standards across asset classes, networks and jurisdictions. The collaboration between CMTA and BfeW works toward this goal by streamlining existing standards and ensuring compatibility between both standardization efforts. Concretely, the project involved analyzing the CMTAT framework against the functions outlined in BfeW's standard and implementing the necessary changes to reflect the requirements of Germany's eWpG and its implementing ordinance eWpRV, as described in the table in section 4. CMTA reviewed the functions of BfeW's standard, which are now covered by the core functions outlined in CMTA's tokenization standards.

Given that tokenization processes must comply with varying legal requirements across jurisdictions, it is important for the technology to be flexible enough to align with different legal orders. CMTAT provides a modular architecture for smart contract design, which allows both scalability and flexibility. Its architecture allows token code to be modified by adding, removing, or adjusting features to align with local regulatory frameworks without compromising structural integrity. Moreover CMTAT is blockchain agnostic, meaning that it is not tied to a particular technology choice and can be implemented using different approaches or technologies that are suitable for the intended use case.

3. TOKENIZATION IN GERMANY

In June 2021 the German eWpG came into force. The law aims to modernize the issuance and trading of securities by making it possible to issue purely electronic securities without a physical document and creating so-called Crypto Securities. The central element of Electronic Securities is their entry into a register, which can only be operated by a BaFin-licensed financial institute. The eWpG allows registration in both centralized and decentralized registers (so-called Crypto Securities Registers)[1]. In 2023 members of the BfeW (then the Bundesverband der Kryptowertpapierregisterführer e.V.) agreed on a technical standard that serves as an appropriate basis for fulfilling the requirements of the German eWpG and eWpRV in managing a Crypto Securities Register. The association recommends using this standard for implementing a token-based register under the eWpG[2].

In order to facilitate implementation of their standard, BfeW supported the idea of creating a common reference implementation which could be easily put to use by issuers. To avoid duplication of efforts and resources, the CMTAT's Solidity reference implementation and standard framework have been modified to match BfeW's requirements.

4. COMPARISON BETWEEN CMTAT AND ERC-7551

A gap analysis between the functionalities outlined in the BfeW standard and the functions of the ERC-7551 and those of the CMTAT framework and CMTAT solidity reference implementation found that the following changes were necessary to make CMTAT consistent with the requirements of German law:

[1] https://www.nyala.de/en/ewpg-en

[2] https://cdn.prod.website-files.com/67ab09799b24477dd9e9573c/67b23140cbe620c976bcdce8_20230915_eWpG_Standard.pdf

- Addition of Forced transfer function (optional)
- Addition of the possibility to partially freeze the balance of a token holder (optional)

These changes were introduced to the CMTAT Framework's Functional Specifications in September 2025 and implemented in the Solidity implementation version 3.0.0 release. A specific deployment version of the CMTAT named "CMTAT ERC-7551" has been created which includes the functions required for issuing tokenized equity securities in Germany.

The below table compares the functionalities and details how the relevant functionalities are implemented in CMTAT's reference Solidity implementation:

N°	**Functionalities**	**ERC-7551 Functions**	**CMTAT v3.0.0**	**CMTAT v3.0.0 Implementation details**	**CMTAT v3.0.0 Modules**
1	Freeze and unfreeze a specific amount of tokens	`freezeTokens` `unfreezeTokens`	☑	Implement ERC-3643 function `freezePartialTokens` and `unfreezePartialTokens` (with and without a `data` parameter) & ERC-3643 function `setAddressFrozen` (with and without a `data` parameter)	EnforcementModule (core) ERC20EnforcementModule (extensions)
2	Pausing transfers The operator can pause and unpause transfers	`pauseTransfers`	☑	Implement ERC-3643 functions `pause/unpause` & `deactivateContract`	PauseModule (core)
3	Link to off-chain document Add the hash of a document	`setPaperContractHash`	Equivalent functionality	The hash is put in the field `Terms` Terms is represented as a Document (name, uri, hash, last on-chain modification date) based on ERC-1643.	
			Function	`setTerms(bytes32 hash, string calldata uri)`	ERC7751Module (options)
			Function	`setTerms(IERC1643CMTAT.DocumentInfo calldata terms_)`	ExtraInformationModule (extensions)
4	Metadata JSON file	`setMetaDataJSON`	☑	Function `setMetaData(string calldata metadata_)`	ERC7751Module (options)
5	Forced transfers Transfer `amount` tokens to `to` without requiring the consent of `from`	`forceTransferFrom`	☑	Two functions are available: with and without the `data` parameter	
			Functions	`forcedTransfer(address from, address to, uint256 value, bytes calldata data)`	ERC20EnforcementModule (extensions)

N°	Functionalities	ERC-7551 Functions	CMTAT v3.0.0	CMTAT v3.0.0 Implementation details	CMTAT v3.0.0 Modules
				ERC-3643 function `forcedTransfer(address from, address to, uint256 value)`	ERC20EnforcementModule (extensions)
6	Token supply management Reduce the balance of `tokenHolder` by `amount` without increasing the amount of tokens of any other holder	`destroyTokens`	☑	Two functions are available: with and without the `data` parameter, as well as a batch version	
			Functions	`burn(address account,uint256 value,bytes calldata data)`	BurnModule (core)
				ERC-3643 function `burn(address account,uint256 value)`	BurnModule (core)
				`batchBurn(address[] calldata accounts,uint256[] calldata values,bytes memory data)`	BurnModule (core)
				ERC-3643 function `batchBurn(address[] calldata accounts,uint256[] calldata values)`	BurnModule (core)
7	Token supply management Increase the balance of `to` by `amount` without decreasing the amount of tokens from any other holder.	`issue`	☑	Two functions are available: with and without the `data`parameter, as well as a batch version (without `data`)	
			Functions	`mint(address account, uint256 value, bytes calldata data)`	MintModule (core)
				ERC-3643 functions `mint(address account, uint256 value)`and `batchMint(address[] calldata accounts,uint256[] calldata values)`	MintModule (core)
8	Transfer compliance Check if a transfer is valid	`canTransfer()` and a `canTransferFrom()`	☑	Implement ERC-3643 function `canTransfer` as well as a specific function `canTransferFrom`	

N°	Functionalities	ERC-7551 Functions	CMTAT v3.0.0	CMTAT v3.0.0 Implementation details	CMTAT v3.0.0 Modules
			Functions	ERC-3643 function `canTransfer(address from,address to,uint256 value)`	ValidationModuleCore
				`canTransferFrom(address spender,address from,address to,uint256 value)`	ValidationModuleCore

5. CONCLUSION

A key outcome of the collaboration between CMTA and BfeW is the implementation of certain changes to the **CMTAT** Framework and Solidity reference implementation to make it consistent with the requirements of Germany's **eWpG** through alignment with ERC-7551.

In order to align CMTAT with German law, the following changes have been implemented:

- Addition of a force transfer function (optional)
- Addition of the possibility to partially freeze the balance of a token holder (optional)

These changes enable the CMTAT to be used to tokenize securities under Swiss law and now also in Germany's regulatory environment.

All the functions of the **ERC-7551** standard, developed by BfeW, are covered by the core functions outlined in CMTA's tokenization standards for Switzerland.

This technical alignment highlights the adaptability of both frameworks and ensures that tokenization processes can be efficiently implemented across multiple jurisdictions.

Further reading:

- https://github.com/CMTA/CMTAT
- CMTAT Framework: Blockchain agnostic functional specifications of the CMTA Token for the tokenization of financial instruments
- ERC-7551 Crypto Security Token Smart Contract Interface (eWpG)