Route de Chêne 30
1208, Geneva
cmta.ch

CMTAT

Functional specifications for the Swiss law compliant tokenization of securities

January 2022

Table of contents

1. INTRODUCTION

The CMTA standard token for securities (CMTAT) is a digital token framework that enables the creation of so-called "ledger-based securities" pursuant to Article 973d of the Swiss Code of Obligations. The CMTAT was primarily conceived for the tokenization of equity securities. Its modular structure does however also make it suitable for the tokenization of other forms of securities, such as debt instruments or structured products. The CMTAT also includes support functionalities, such as reliable deployment and gasless transactions.

The CMTAT is an evolution of CMTA20, a token that was first released in 2019 as an Ethereum smart contract. CMTA20 was conceived as an instrument allowing Swiss issuers to conform with CMTA's Blueprint for the tokenization of shares of Swiss corporations, which was released in October 2018. Further to the entry into force of the Swiss Federal Act on the Adaptation of Federal Law to Developments in Distributed Ledger Technology of 2020, CMTA has replaced the Blueprint of 2018 with a new Standard for the tokenization of shares of Swiss corporations using the distributed ledger technology, which was adopted in November 2021. The CMTAT has been conceived to reflect the change from the Blueprint of 2018 to the Standard of 2021.

The CMTAT mainly differs from the CMTA20 in the following aspects:

A generic, platform-agnostic, rather than a monolithic, Ethereum-oriented token

Modular design with a core set of functionalities and optional modules

Support for "gasless" transactions on Ethereum (*i.e.* transactions where the fee is charged to a party other than their initiators).

a. Contributors

The CMTAT was developed by a working group created by CMTA under the supervision of CMTA's Technical Committee.

b. Intellectual property

CMTA and CMTA members have not claimed any copyright or other property rights on the computer code on which the CMTAT is based, nor are they aware of third parties having claimed such rights. The copyright to the CMTAT is owned by the CMTA. The code of the CMTAT is released under Mozilla Public License 2.0.

c. Disclaimer

Pursuant to the Mozilla Public License 2.0, the code of the CMTAT is provided on an "as is" basis, without warranty of any kind, either expressed, implied, or statutory, including, without limitation, warranties that such code is free of defects, merchantable, fit for a particular purpose or non-infringing. The entire risk as to the quality and performance of the CMTAT code is with the user thereof. Should the CMTAT code prove defective in any respect, the relevant user (not the CMTA or any contributor) will assume the cost of any necessary servicing, repair, or correction.

2. OUTLINE OF THE CMTAT FRAMEWORK

a. Basis

The functionalities of the CMTAT framework draw on Appendix 2 to the CMTA's Standard for the tokenization of shares of Swiss corporations using the distributed ledger technology of November 2021, which describes the functionalities that are to be used to represent equity securities. The CMTAT framework may however be supplemented to include functions that are specific to other forms of securities, such as debt instruments or structured products.

This document describes the functions that may be used (a) to create tokens that are fit to represent securities in accordance with Swiss law and (b) to manage certain events affecting the securities using on-chain or off-chain mechanisms, where appropriate.

b. Modular structure and other key changes

The CMTAT framework offers flexibility by dividing the various functionalities of the framework into modules. Aside from the base module (described below), all modules are optional. Certain functionalities of a particular module may also be optional, as described below.
The CMTAT features a number of changes compared to the CMTA20. In particular, the CMTAT:

- incorporates changes to account for the recently adopted Swiss legislation;
- includes a generic, platform-agnostic description of modules composing the token standard;
- describes deployment models to enable stronger governance procedures;
- supports multiple roles (issuer, delegate, token holder, as well as function/module-specific roles);
- is blockchain-agnostic and at the time of publication includes a reference implementation for the Ethereum Blockchain. Implementations can be developed for other blockchains, such as Tezos; and
- supports "gasless" transactions on Ethereum.

CMTAT's modular structure makes it possible to add features that can be used to support certain corporate actions of the issuer of the relevant tokenized securities, such as dividend distributions or interest payments. The base version of the CMTAT does however not include such features. Likewise, CMTAT could be supplemented to introduce features that would make it possible to condition the transfer of tokens to the consent of its issuer, thereby making it possible to support share transfer restrictions (so-called "Vinkulierung") for Swiss stock corporations (Aktiengesellschaften, sociétés anonymes) whose shares are not listed on a stock exchange within the meaning of Swiss corporation law (Articles 685b et seq. of the Swiss Code of Obligations). The base version of the CMTAT can however be used to tokenize shares of Swiss stock corporations whose articles of association contain transfer restrictions, but whose shares are listed on a Swiss stock exchange (Articles 685d et seq. of the Swiss Code of Obligations).

c. Terminology and color code

In this document:

- "issuer" means the legal entity that has issued the securities represented by the tokens or a person or entity authorized by such issuer to perform certain actions on the tokens; and
- "user" means (absent any precision) any entity or person that controls an address on the distributed ledger on which the CMTAT tokens are recorded (but does not necessarily hold CMTAT tokens).

The functions of the CMTAT framework are identified as follows:

- functions marked with a purple square (■) are functions that may be used by the issuer; and
- functions marked with an azure dot (●) are functions that may be used by any user.

3. TOKEN FUNCTIONAL MODULES

This section describes the functionalities of the CMTAT framework, and groups these functionalities into modules.

The list of functionalities in each module description is the minimal set of functionalities that must be implemented. Additional platform-specific functionalities may be required for compliance with standard APIs. For example, the `approve()` and `transferFrom()` ERC-20 functions are not listed, but are implemented in the reference Ethereum version. Also, some blockchains may require specific implementation strategies for certain functionalities, such as transfer validation rules.

For an example of implementation, including platform-specific application programming interfaces (APIs), events emitted, and other aspects, please refer to the reference implementation at https://github.com/CMTA/CMTAT

a. Base module (Mandatory)

CMTAT's base module contains the basic functionalities that a token must have to be associated with an equity security or any other form of security. The base module provides the functionality that is essential for a CMTAT token to be a fungible token circulating on a blockchain.

Functionalities:

1. **TotalSupply**: For a particular CMTAT token, any person may know the total number of tokens in circulation at any point it time. ●

2. **BalanceOf:** For a particular CMTAT token and a particular user, any person may know the number of tokens currently recorded on the user's ledger address. ●

3. **Transfer**: Users may transfer some or all of their tokens to some other ledger address (that the transferor does not necessarily control). ●

4. **Mint:** Issue a given number of tokens to a given ledger address. ■

5. **Burn:** Burn (destroy) a given number of tokens from a given ledger address. ■

6. **Pause:** Prevent all transfers of tokens on the ledger until **"Unpause"** is called. ■

7. **Unpause:** Restore the possibility to transfer tokens on the ledger, in principle after **"Pause"** is called. ■

8. **Kill:** Self-destruction of the contract and effectively of the tokens, thereby preventing any transfer or other operation. ■

Mandatory attributes, applicable to all CMTAT tokens:

- Name
- Ticker symbol (optional)
- Token ID (ISIN or other identifier) (optional)
- Reference to the terms of tokenization, the terms of the instrument, and other relevant documents (e.g. prospectus or key information document). The reference can take the form of an URL, a combination of an URL and of specific directions allowing the user to retrieve the relevant documents (e.g. "[domain].com/shares > Tokens") or a fingerprint.

Note that decimals number must be set to zero (which means that the tokens admit no fractional parts).

Optional attributes, applicable to tokens used for debt securities:

- Guarantor identifier (if any)
- Bondholder representative identifier (if any)
- Maturity date
- Interest rate
- Par value (principal amount)
- Interest schedule format (if any). The purpose of the interest schedule is to set, in the parameters of the smart contract, the dates on which the interest payments accrue.
    - Format A: start date/end date/period
    - Format B: start date/end date/day of period (*e.g.* quarter or year)
    - Format C: date 1/date 2/date 3/….
- Interest payment date (if different from the date on which the interest payment accrues):
    - Format A: period (indicating the period between the accrual date for the interest payment and the date on which the payment is scheduled to be made)
    - Format B: specific date
- Day count convention
- Business day convention
- Public holidays calendar

b. Snapshot module (Mandatory)

Rationale: In relation to distributions or the exercise of rights attached to tokenized securities, it is necessary to determine the number of tokens held by certain users at a certain point in time to allow issuers to carry out certain corporate actions such as dividend or interest payments. Such moments are generally referred to in practice as the "record date" or the "record time" (i.e. the time that is relevant to determine the eligibility of security holders for the relevant corporate action). The snapshot functions to determine the number of tokens recorded on the various ledger addresses at a specific point in time and to use that information to carry out transactions on-chain.

When tokenized securities are actively traded, the actual ownership of securities may be different when the corporate action is executed (e.g. the time when the dividend or interest is actually paid) compared to when the right of the relevant security holders arose (e.g. the time when the dividend or interest became due). The Snapshot module makes it possible to carry out transactions (either on-chain or off-chain) based on the state of the ledger at the record time rather than at the moment the corporate action is executed. The function makes it possible to execute a particular action simultaneously at a time determined in advance, in a manner that guarantees that each token benefits once (but only once) from the corporate action.

Functionalities:

1. **ScheduleSnapshot**: For a particular CMTAT token, the issuer may schedule the creation of a snapshot at a certain time. The time of the newly scheduled snapshot cannot be before the time of the latest scheduled, but not yet created, snapshot. ■

2. **RescheduleSnapshot:** The issuer can change the time of a scheduled snapshot. The new scheduled time cannot be before the time of the previously scheduled snapshot or after the time of the next scheduled snapshot (*i.e.* scheduled snapshots cannot be reordered). ■

3. **UnscheduleSnapshot**: For a particular scheduled snapshot, the issuer can cancel a previously scheduled snapshot. The unscheduled snapshot must be the last scheduled snapshot, and its time must be in the future. ■

4. **SnapshotTime**: For a particular scheduled, but not yet created, snapshot, anyone may know the snapshot time.

5. **SnapshotTotalSupply**: For a particular created snapshot, anyone may know the total number of tokens that were in circulation at the snapshot creation time.

6. **SnapshotBalanceOf**: For a particular created snapshot and a particular ledger address, anyone may know the number of tokens recorded on the relevant ledger address at the snapshot creation time.

c. Validation module

Rationale: Issuers may decide to implement legal restrictions to the transfer of the tokenized instruments, to limit the scope of persons or entities who may hold the relevant instruments. The Validation module allows this by providing that all transactions (except for forced ones) require the pre-approval of the issuer. These restrictions may be typically implemented for tokenized shares where the issuer has implemented transfer restrictions (so-called "Vinkulierung" provisions) in its articles of association and where the relevant shares are not listed on a stock exchange (within the meaning of Article 685b et seq. of the Swiss Code of Obligations).

Functionalities:

1. **ValidateTransfer**: Send a request for validation of a particular transfer, given the sender and recipient addresses, and the amount to the issuer. ●

2. **SetRuleEngine**: Assign a set of rules to be enforced by the **ValidateTransfer** function. Said rules are to be defined in a separate contract. ■

d. Authorization module

Rationale: Issuers may wish to implement a role-based access control to the token functionalities, rather than distinguish only between issuer and user. This may help reflect the issuer organization's governance model. The Authorization module thus allows the issuer to assign responsibilities and authorizations to various persons (accounts).

Functionalities:

1. **GrantRole**: Grant a role to a given account. ■

2. **RevokeRole**: Revoke a role from the given account. ■

3. **HasRole**: Tell whether a given account has a given role. ■

4. ADDITIONAL GUIDELINES

This section provides guidance as to how certain corporate actions can be managed when CMTAT is being used. These are not part of the current reference implementation, and are provided for information purpose only.

a. Share splits and reverse splits

The CMTAT does not make it possible to manage share splits and reverse splits on-chain (i.e. to redefine the notion unit of a token). Rather, such corporate actions must be carried out through the "Mint" and "Burn" functions described above, to ensure that the number of tokens always remains equal to the number of tokenized shares.

A share split can be carried out by either "Burning" the existing tokens and "Minting" new ones (e.g. by "Minting" two new tokens for each token "Burnt") or by "Minting" additional tokens and allocating such new tokens to all the ledger addresses. Share splits and reverse splits require amendments to the relevant issuer's constitutive documents (articles of association and potentially the tokenization terms).

b. Distribution of dividends or interest

Issuers may be required to make distributions to the securities' holders (e.g. dividend payments for equity securities or interest payments for debt securities). Issuers may wish to carry out distributions off chain (i.e. by transferring fiat currencies to the securities' holders' bank account). However, if the issuer intends to carry out such distributions on-chain, this may require the distribution of new tokens to existing token holders, on the basis of a snapshot carried out at the moment the legal entitlement to the distribution arises. Not all token holders may be eligible for distributions. For example, under the CMTA's *Standard for the tokenization of shares of Swiss corporations using the distributed ledger technology*, only token holders that have identified themselves in a satisfactory manner to the issuer are eligible to receive distributions or exercise other shareholder rights.

Distribution events are typically performed according to a predefined schedule, and according to the token distribution at a given time, which can be determined by a snapshot performed without the Snapshot module.

A module to manage distribution on-chain may support the following functionalities:

1. **DistributionCreateParameters**: Define settlement token (*i.e.* the token that is to be distributed), identify a (past or future) block time/height for distribution snapshot, and amount to be distributed. ■

2. **DistributionSetEligibility**: Flag a given users' tokens as being eligible or non-eligible to receive distributions (default: eligible). ■

3. **DistributionSetDeposit**: Send deposit amount for claiming settlement tokens to token holders flagged as eligible. ■

4. **DistributionClaimDeposit**: Allow token holders to claim their share of a deposit, identified by a deposit Identification, according to the token balance at the snapshot created at the defined time/height. ●

Additional use cases for tokens representing debt instruments:

5. **DistributionSchedule**: Define a schedule for interest payments [and repayment of the par value at maturity], based on the token attributes. ■

6. **DistributionUnschedule**: Cancel the previously set schedule for interest payments [and repayment of the principal amount at maturity].■

c. Credit events

A number of events may occur during the lifetime of a debt instrument. Contrary to corporate actions, some of these events typically involve third parties (e.g. a bond agent or a rating agency). The issuer may in such cases wish to delegate the relevant functions to the relevant third parties.

A module to manage on-chain credit events may support the following functionalities:

1. **FlagDefault**: The tokens are flagged as representing debt instruments in respect of which the issuer has defaulted (bond agent function). ■

2. **FlagDefaultRemove**: The tokens are no longer flagged as representing debt instruments in respect of which the issuer has defaulted (bond agent function). ■

3. **FlagRedeemed**: The tokens are flagged as representing debt instruments that have been redeemed (bond agent function).■

4. **SetRating**: The tokens are flagged as representing debt instruments that have been given a particular rating (bond agent). ■

5. REFERENCE IMPLEMENTATION

A reference implementation for the Ethereum platform, in the Solidity language, is available at https://github.com/CMTA/CMTAT, as an ERC-20-compatible token, following a modular design. ERC-20 defines the standard API for fungible tokens circulating on the Ethereum blockchain, as supported by Ethereum-based wallets, DeFi applications, and layer-2 protocols, for example.

This CMTAT implementation includes platform-specific features regarding deployment, upgradeability and security aspects, namely:

- Upgradeability of the token contract, thanks to support for deployments via proxy contracts.
- Support for transaction fee transfer, also known as "gasless transactions" in the Ethereum context.
- Definition or functionality-based roles for simplified governance management.