International Capital Market Association

ICMA

Summary of third-party proposals on integration of ICMA's Bond Data Taxonomy in DLT platforms for Project Guardian

25 October 2024

Introduction

This publication outlines three third-party conceptual proposals for the integration of the ICMA's Bond Data Taxonomy (BDT) into DLT bond workflows. The proposals are set out by Tokeny, UBS, and FeverTokens.

Tokeny's propsal represents the Bond Onchain by using a compliant-by-design token framework such as ERC 3643 combined with the BDT to describe the asset, allowing the automation of both the issuance process and the post trade lifecycle management.

The proposal by UBS demonstates how to augment the Capital Markets and Technology Association (CMTA) smart contract framework for the tokenisation of debt securities under Swiss law, with BDT fields and XML schema, such as issuer name and ticker. This can make financial transactions faster, more secure and enhance interoperability.

FeverTokens' proposal sets out how the BDT can be integrated in a modular approach into ERC token standards, through a separate solidity-based package and linking to off-chain data. The steps for which are set out in the FeverTokens section below, including the integration with ERC 20 and ERC 1400 Token Contracts.

Project Guardian

The Monetary Authority of Singapore's Project Guardian aims to drive and scale asset tokenisation in fixed income, foreign exchange (FX) and asset & wealth management. This included partnerships with global industry associations and financial institutions.

In Project Guardian's fixed income workstream, ICMA's BDT is built upon to promote industry standards, fixed income protocols and technical data specifications for asset tokenisation (DLT-based). The aim is to provide guidelines that facilitate the use of different protocols and smart

contracts based on a consistent data model across the securities lifecycle – both for digital (DLT-based) bonds and traditional debt securities.

The Project Guardian publication in full can be found here.

ICMA Bond Data Taxonomy

Whilst the tokenisation of assets is a relatively new phenomenon, the core foundations of the debt security remains, regardless of its form. For example, the econmic terms such as an issuance amount denominated in a currency, maturity date and ISIN, remain irrespective of a ”tokenised” or ”traditional” bond security. ICMA's BDT, launched in early 2023, provides a common language for key bond information typically contained in a bond term sheet in a (i) standardised and (ii) machine-readable (XML) format. The BDT defines unambiguously the data elements of a debt security as well as related information such as governing law, selling restrictions, and DLT platform information that may be communicated between different parties in a transaction (e.g. via APIs, smart contracts, or spreadsheets).

From a technical perspective, each field is defined by its data type, multiplicity and restrictions in the BDT's XML Schema Definitions (XSDs). Many of the fields are defined by ISO standards such as date, currency and legal identifiers. The XSDs' embedded logic includes syntax patterns and predefined enumerations to facilitate accurate and consistent data exchange and straight-through processing (STP), for example, for the announcement of a new issue, allocation and pricing, clearing and settlement, distribution or asset servicing.

The BDT has been designed and developed by a broad range of bond market constituents to promote automation and reduce the risk of fragmentation across issuance, trading, settlement and distribution of debt securities. The BDT is technology agnostic and designed to be used both for traditional debt securities as well as DLT-based debt securities.

Disclaimer

ICMA does not recommend or endorse any third-party solution or implementation proposal.

Tokeny Proposal for Extension of ERC3634 for ICMA Bond Data Taxonomy

Overview

Representing the Bond Onchain using a compliant-by-design token framework such as ERC 3643 combined with the BDT to describe the asset allows the automation of both the issuance process and the post trade lifecycle management, with the following key benefits:

- Early security/ISIN creation due to early receipt of deal information by CSD. This also allows for the creation of the security in downstream systems.
- Auto-generation of settlement instructions: The auto-generation of settlement instructions reduces the operational burden; this does, however, require a shift to messaging generation by centralised market infrastructure such as CSDs to arranger and investor custodians.
- Settlement time reduction: New issue settlement is currently T+5 due to processes such as custody approval, deal documentation, settlement instruction generation and investor sub-allocations. Lead times for approval and completion of post-trade processes can be significantly reduced.
- Elimination of operational risk as a result of collapsing the issuance flows into a single atomic transaction.
- Independence from batch-driven settlement cycles: The bond's settlement can have no dependency on the CSD's batch settlement cycle given the settlement will be triggered upon completion of conditions precedent.
- Asset servicing automation: such automation allows paying agents and trustees to focus on higher value-added activities for issuers and clients.

Project Long Term Vision

The project long term vision is to build an *Onchain Bond platform* to streamline and automate the workflows between the Issuer, Placement Agent, Paying Agent, Custodian, Clearing House and Investor.

ERC 3643 Token Features

- Base Token Specification
- Transfer Restriction
- White List Management
- Token Contract Pause
- Third party Security Audit

- Role Based Access Control (Agent)
- Mint and Burn to Any Address
- Forced Transfer Function
- Contract Version Tracking
- License: GPL 3.0

Objective

- **Build and validate** the functional requirements & user journey(s) of an *Onchain Bond Platform* interoperable with various vendors services
- **Compare Issuance processes** on the *Onchain Bond Platform* versus current market practices and measure gains
- **Compare Asset Servicing process** on the *Onchain Bond Platform* versus current market practices and measure gains
- Validate the **technical and commercial feasibility**
- Understand **non-functional requirements for go-live**

Project Scope

Participants

The project will bring together various stakeholders of the debt market, including:

- Issuer
- Issuer Manager
- Placement Agent
- Institutional Investor
- Paying Agent
- Custodian
- Clear House

In the framing phase (phase 1) of the project, we will need to identify partners to cover each of these role in order to map their responsibilities in the blockchain world and clearly define their respective roles and permissions on the Onchain Bond Platform. The ERC-3643 framework allows full flexibility in terms of permissions of authorized BDT roles and actions that they can perform.

Financial Instrument

- In the framing phase (phase 1) of the project, we will define 2 or 3 instruments to tokenize in a test environment.
- The instruments can be any type of bond covered by ICMA's **Bond Data Taxonomy (BDT**).
- It will be registered on the blockchain using the compliant **ERC 3643 token standard.**

Use-case workflows

The process for issuance will be as follows :

- Token and AssetID deployment :

  The Issuer, or it's manager, will be responsible for deploying the Token representing the Bond using the permissioned ERC-3643 standard and deploying the linked AssetID smart contract automatically, using the standardized Taxonomy on the Onchain Bond Platform; Alternatively, the process of deploying the Token and the AssetID could take place directly at the CSD (responsibilities to be defined in the framing phase)
  The total issuance amount (the cap) will be encoded in the token to ensures that no more securities can be issued than initially agreed, preventing any oversupply;

- ISIN Code generation

Once the legal documents are finalized, this information is then shared automatically with the Central Securities Depository (CSD) to generate the ISIN code. The CSD acts as an oracle, updating the AssetID with the ISIN.

- Order Management

  The Onchain Bond Platform allows market participants to submit purchase instructions prior to the closing date via a Delivery Versus Payment (DvP) transaction. Additionally, the buyer's funds can be blocked in their wallet, ensuring they are available for the transaction with the same value date; On the closing date, the CSD will create the securities in the issuer's wallet (mint tokens), matching the previously submitted purchase instructions; The securities can then be delivered either to the custodian appointed by the investor or directly into the investor's wallet.

- Lifecycle Management

  The process for the bond lifecycle management will be as follows:

  The paying agent will log in to the Onchain Bond Platform to sign transactions on the automatically calculated distributions; Alternatively, we can test the stripping of principal and coupon payment by providing a smart contract that transforms 1 Bond token into 1 Strip Coupon token and 1 Strip Principal token.

Integration of BDT into DLT Workflows

STEP 1: Bond setup phase

The setup phase will involve the automatic deployment of the three Smart Contracts described above:

- Token
- AssetID
- Bond Vault Smart Contract

Extracts of the BDT data will be needed to generate both the ISIN and the three Smart Contracts representing the asset (Token, ASSETID and Bond Vault Smart Contract)

The ISIN can be either generated before or after the BDT is used to deploy the AssetID smart contract, as partial smart contract state is allowed: AssetID can be deployed as an empty container linked to the token and data can be added / removed dynamically by the Agent appointed by the owner. As mentioned above, in phase 1 of the project we will need to detail which entity can take this Agent role and manage their specific permissions to tailor the operational flow.

STEP 2 Issuance/Distribution phase

Details on the specific business logic will be fine-tuned during the framing phase of the project, yet we expect the following flows:

1. Issuer configures distribution through BDT/AssetID: Accepted onchain payment currency, price, min/max investments, closing date (if any), etc.
2. The investor places an order. Order is registered. Onchain payment funds are blocked on investor wallet by the vault smart contract.
   Note: Alternatively, payment funds can be transferred to bond vault.
3. Execute DvP. Bond vault smart contract executes DVP in an atomic way (in a single transaction) by:
   a. Transfer payment funds from investor wallet to the vault. This requires previous authorization from the investor.
   b. Mint bond tokens from bond vault to investor wallet. Alternatively, bond tokens could be minted to the bond vault at setup (phase 0) and transferred to investor wallet here

   Execution can be triggered by the issuer or automatically at close date.

Distribution data table (BDT):

Specific business logic based on this data to be considered during implementation phase.

The following data will be taken from the BDT file to populate the AssetID attributes:

Specified Denomination
Specified Currency
Payment Currency
Settlement Currency
Pricing Date
Issue Date
Settlement Date
Issue Price
Fees
FeeAllocationType
StartDate
EndDate

STEP 3: Coupon/Payment phase

Details on the specific business logic will be fine-tuned during the framing phase of the project, yet we expect the following flows:

1. Issuer configures coupon payment through BDT/AssetID.
2. Execute coupon payment. Bond vault transfers payment coupon funds from issuer(paying) agent wallet to investor wallet based on each investor bond token balance.
   Execution can be triggered by the issuer(paying) agent or automatically based on defined periodicity.

   Payment coupon funds could also previously be transferred to bond vault from issuer(paying) agent wallet and transferred to investor from there.

If needed, the CSD could interact with the Bond Vault smart contract to modify contract parameters related to corporate actions. The agent interactions should be configured at function level. CSD could be responsible for parameterizing the actions and another agent of executing the actions (but not allowed to change parameters). This will need to be clarified during phase 1 of te project.

Coupon payment data table (BDT):

Specific business logic based on this data to be considered during implementation phase.

The following data will be taken from the BDT file to populate the AssetID attributes:

Interest Type
Interest Payment Frequency
Term
Term Period
Calculation Type
Determination Method
Observation Method
Period
Interest Commencement Date
First Interest Payment Date
BrokenDateType
Last Interest Payment Date
PayableDate

STEP 4: Redemption phase

Details on the specific business logic will be fine-tuned during the framing phase of the project, yet we expect the following flows:

1. Execute redemption. Bond vault smart contract executes redemption in an atomic way (in a single transaction) by:
    a. Block bond tokens to redeem
    b. Transfer redemption payment from issuer agent wallet to investor wallet
    c. Burn bond tokens on investor wallet.

    Execution can be triggered by the investor or automatically based on maturity date.

    Redemption funds could also previously be transferred to bond vault from issuer(paying) agent wallet and transferred to investor from there.

Redemption data table (BDT):

Specific business logic based on this data to be considered during implementation phase.

The following data will be taken from the BDT file to populate the AssetID attributes:

Specified Denomination
Final Redemption Amount
Early Redemption Amount
Specified Currency
Payment Currency
Settlement Currency
Pricing Date
Issue Date
Settlement Date
Maturity Date
Issue Price
Fees
FeeAllocationType
Optional Redemption Dates
StartDate
EndDate
Redemption Payment Basis
Optional Redemption Amount

Blockchain Network

The bonds can be registered on any EVM-compatible blockchain network. The choice of network will be determined during the framing phase of the project.

Token Design

The token will be implemented as a **T-REX permissioned token (ERC-3643)**,

Asset Design

The Asset Attributes will be defined using the **ICMA's Bond Data Taxonomy (BDT)**.

Part or all of these metadata can be represented On-chain via the Asset OnchainID, depending on the use-case to be clarified in the Framing phase of the project. Based on the assumptions presented in section 2.1, the BDT data fields to be recorded onchain would be the following:

Specified Denomination
Specified Currency
Payment Currency
Settlement Currency
Pricing Date
Issue Date
Settlement Date
Interest Type
Interest Payment Frequency
Term
Term Period
Calculation Type
Determination Method
Observation Method
Period
Interest Commencement Date
First Interest Payment Date
BrokenDateType
Last Interest Payment Date

PayableDate
Issue Price
Fees
FeeAllocationType
StartDate
EndDate
Interest Type
Interest Payment Frequency
Term
Term Period
Calculation Type
Determination Method
Observation Method
Period
Interest Commencement Date
First Interest Payment Date
BrokenDateType
Last Interest Payment Date
PayableDate

Project Phases

Phase 1 - Framing

During the framing phase we will:

- Identify partners covering each bond stakeholders
    - Issuer
    - Issuer Manager
    - Placement Agent
    - Institutional Investor
    - Paying Agent
    - Custodian
    - Clear House
- Clearly defined roles and responsibilities of each participant on the Onchain Bond Platform
- Determine the pilot Blockchain Network
- Define the pilot Financial Instruments to tokenize

Phase 2 - Developments

Developments will be made based on the Tokeny T-REX platform leveraging the ERC-3643 for faster time to market

Phase 3 - Testing and Learning

Objectives are to

- **Compare Issuance processes** on the *Onchain Bond Platform* versus current market practices and measure gains
- **Compare Asset Servicing process** on the *Onchain Bond Platform* versus current market practices and measure gains
- Validate the **technical and commercial feasibility**
- Understand **non-functional requirements for go-live**

UBS Proposal to Augment CMTAT with ICMA Fields and Tamper-Proof Methodology

Overview

The Capital Market Technology Association Token (CMTAT) is a digital token standard developed to streamline the issuance, trading, and management of securities on blockchain platforms. It aims to improve efficiency, security, and compliance in capital markets by providing a robust framework for tokenizing financial instruments like bonds and equities. Integrating fields proposed by the International Capital Market Association (ICMA) into CMTAT will ensure industry-standard adherence and enhance the token's utility for bond issuance.

Integrating ICMA fields into CMTAT is crucial for standardizing the capital markets industry. Using a common taxonomy like the Bond Data Taxonomy (BDT) and XML schema for transaction details enables faster, safer exchanges of financial instruments. This standardization enhances interoperability, streamlines collaboration, reduces errors, and ensures regulatory compliance, leading to a more efficient, transparent, and secure capital markets ecosystem.

CMTAT Token Features

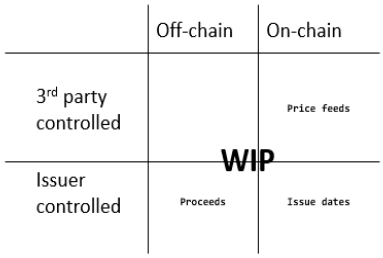
- Base Token Specification
- Transfer Restriction
- Document Management
- White List Management
- Token Contract Pause
- Snapshot/Checkpoints
- Support for Debt Instruments
- Thirdparty Security Audit
- Role Based Access Control
- Security Identifiers
- Mint and Burn to Any Address
- Gasless Support (ERC 2771)
- Customizable Modular Design
- License: MPL 2.0

On-chain vs Off-chain BDT Fields

The Capital Market Technology Association considers two categories of fields in the BDT:

– Those whose values are meaningful for on-chain like issue dates pricing fees etc

– Values that aren't needed for any on-chain operations like Proceeds, Ticker etc



Additionally, some fields are the responsibility of the issuer, while others can benefit from a common distributed ledger and a robust taxonomy. For instance, fields managed by a distributed ledger could lead to significant industry improvements, such as a future scenario where a rating agency automatically rates a bond on the blockchain. This integration not only enhances transparency and efficiency but also ensures that critical data is securely and accurately managed across the industry.

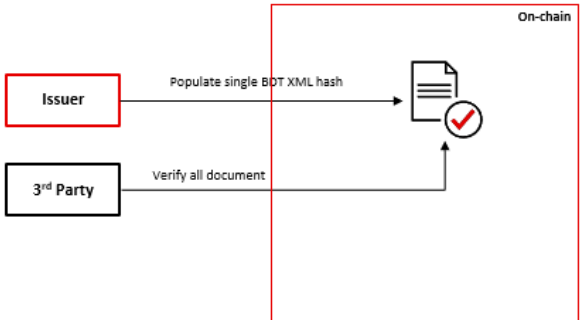
Modular Design Proposal

- We propose a modular approach to contract design, like the pattern used in CMTAT. By utilizing abstract contracts for each Key Fields Parent Group, we can create a flexible and scalable architecture.
- These abstract contracts can then be included in the debt standard, ensuring that only the fields that must be on-chain are incorporated.
- This modular design allows issuers to compose bonds in the most appropriate way, leveraging the benefits of blockchain technology while maintaining a robust and adaptable framework.

Tamper Proofing Methodology

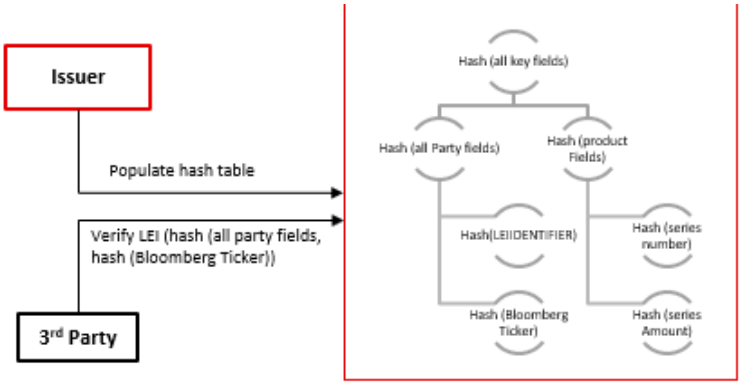
Trivial solution based on a single hash of the bond data taxonomy, pros easy to implement& maintain, cons the entire content of the bond needs to be reveled.

- We propose leveraging the new document module from the dev branch of CMTA to enhance tamper proofing.
- By hashing the bond XML and adding it into the document field, we can ensure the integrity and authenticity of the bond data.

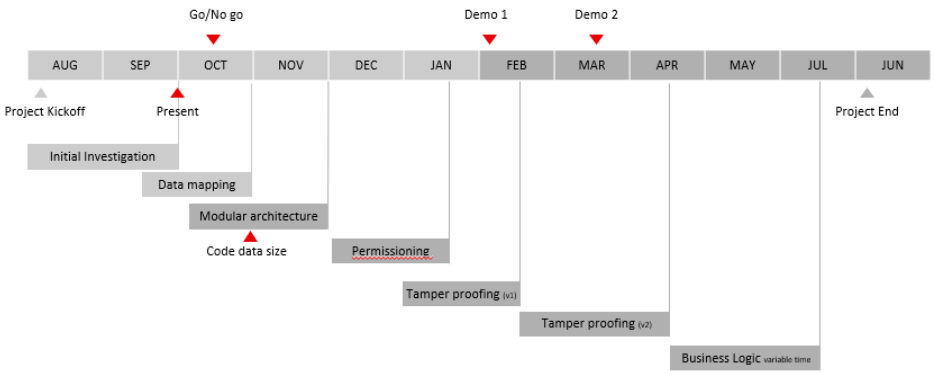


Complex solution based on a tree representation of the bond data taxonomy, pros need to know reveal of BDT fields, cons additional complexity and maintaining cost.

- Additionally, a more advanced solution could involve using a Merkle tree to tamper-proof each Key Fields Parent Group.
- This approach would provide a cryptographic guarantee that each field remains unaltered, thereby enhancing security and trust for all stakeholders involved in the bond issuance process.



Proposed Timeline for the Implementation of the Dual Standard Token Based on a Modular Design and Tamper-Proof Mechanism



Notes:

- This timeline excludes further steps such as audits, legal reviews, compliance etc.
- Given the standardized nature of the Bond Data Taxonomy (BDT), only the business logic may require re-coding for each new product.
- Potential Challenges:
  - Solidity code size constraints

FeverTokens Proposal for Integration of ICMA Bond Data Taxonomy

Overview

FeverTokens proposes integrating the BDT into DLT platforms through a separate solidity-based package and linking to off-chain data.

BDT Package Integration Outline

1. Understanding the Bond Data Taxonomy (BDT)

- Reviewing ICMA's Bond Data Taxonomy to identify all relevant fields.
- Understanding the regulatory and operational importance of each field.

2. Categorizing BDT Fields: On-Chain vs. Off-Chain

- On-Chain Criteria:
    - Essential for Compliance: Critical identifiers like LEIs and ISINs.
    - Immutability: Data that does not change, such as issuance dates or interest rates.
    - Frequent Access: Data that the smart contract regularly interacts with, like interest rates or redemption amounts.

- Off-Chain Criteria:
    - Large Data: Descriptive text fields, legal documents, or lengthy restrictions.
    - Sensitive Information: Business-sensitive data, such as issuer names or detailed use of proceeds.
    - Frequently Updated: Data likely to change over time, such as ratings or outlooks.

- Example:
    - On-Chain: `Issuer LEI`, `Series Amount`, `Maturity Date`

- Off-Chain: `Issuer Name`, `Use of Proceeds`, `Guarantor Ratings`

3. Defining Solidity Names and Data Types for On-Chain Data

- Naming Convention:
    - Using camelCase for Solidity variable names.
    - Prefix with the relevant context (e.g., `issuerLEI`, `maturityDate`).

- Solidity Data Types:
    - `string`: For text and identifier fields.
    - `uint256`: For numeric values such as amounts, interest rates.
    - `bool`: For true/false fields like DLT indicators.
    - `address`: For Ethereum addresses if applicable.

- Example:
  - `string public issuerLEI;`
  - `uint256 public seriesAmount;`
  - `string public maturityDate;`

4. Designing Data Structures for Diamond Storage

- Diamond Storage Pattern:
    - Creating data structs (combination of structures and mappings) in the Diamond Storage to handle all BDT fields.
    - Using the Diamond Storage to store the data in a fixed storage location accessible by the BDTPackage.

- Example:

```solidity
```

```
struct BondData {
    string issuerLEI;
    uint256 seriesAmount;
    string maturityDate;
    uint256 interestRate;
    bool dltBondIndicator;
    // ...
}

struct Layout {
    BondData bondData;
}

uint256 public constant STORAGE_SLOT = keccak256("diamond.standard.bondData");


function layout() internal pure returns (Layout storage ds) {

    assembly {
        ds.slot := STORAGE_SLOT
    }
}
```

5. Implementing the BDTPackage

- Purpose: Handling all interactions with the Bond Data within the token.

- Functions:

- Setter Functions: Functions to set each on-chain BDT field.

- Getter Functions: Functions to retrieve each on-chain BDT field.

- Example:

```solidity
contract BDTPackage {
    function setIssuerLEI(string memory _issuerLEI) external {
        DiamondStorage storage ds = diamondStorage();
        ds.bondData.issuerLEI = _issuerLEI;
    }

    function getIssuerLEI() external view returns (string memory) {
        DiamondStorage storage ds = diamondStorage();
        return ds.bondData.issuerLEI;
    }
}
```

6. Linking Off-Chain Data

- IPFS/Oracles:

    - Storing large or sensitive data off-chain using IPFS or another decentralized storage solution.

    - Using oracles to link off-chain data to the smart contract, ensuring data integrity.

    - On-Chain References:

    - Storing a hash or reference (e.g., IPFS CID) on-chain to ensure verifiable linkage to the off-chain data.

- Example:

- `string public ipfsHashOfIssuerName;`

- A method to validate off-chain data using the stored hash.

7. Integration with ERC-20/ERC-1400 Token Contracts

- ERC-20/1400 Compliance: Ensure that the BDTPackage interacts seamlessly with existing ERC-20 or ERC-1400 functions.

- Extensions: Adding additional functions specific to bond-related actions like interest payments, redemption, etc.

- Integration: Adding BDTPackage as a package (facet) in the Diamond Standard structure alongside other packages for ERC-20/ERC-1400 compliance.

8. Testing and Deployment

- Testing on Testnet: Deploying the BDTPackage on a test network to ensure all functionalities work as expected.

- Security Audits: Conducting thorough audits to ensure the integrity and security of the smart contract.

- Mainnet Deployment: Once tested and audited, deploying to the mainnet.

9. Documentation and Compliance

- Documenting the Integration: Clearly document the integration process, including how off-chain data is handled.

- Compliance Checks: Ensuring all required regulatory compliance checks are in place, especially for bond-related data.

Disclaimer

ICMA does not recommend or endorse any third-party solution or implementation proposal.