Whitepaper

ERC3643

The T-REX protocol *(Token for Regulated EXchanges)*

The token standard for real-world asset tokenization

Version 4.0 - May 23rd, 2023

The source code is available under GPL 3.0 license on Github: https://github.com/TokenySolutions

TABLE OF CONTENT

Contributors:
Joachim Lebrun,
Luc Falempin,
Kevin Thizy,
Tony Malghem,
Xavi Aznal,
Thaddee Bousselin,
Fabrice Croiseaux

Executive summary

The advent of Bitcoin and other cryptocurrencies sparked an initial wave of Initial Coin Offerings (ICOs), leveraging Distributed Ledger Technology (DLT) for issuing diverse digital assets in the form of utility tokens[1], which demonstrated the potential of blockchain as a shared infrastructure for transferring assets.

By directly managing tokens in their wallets, users could transfer the token ownership peer-to-peer without intermediaries, bringing unprecedented efficiency, accessibility, and liquidity to the cryptocurrency market.

Financial markets, especially private markets, are craving the same level of efficiency, accessibility, and liquidity. The catch is that tokenized securities, or security tokens cannot be permissionless tokens like utility tokens, which can be transferred to anyone. They must be permissioned tokens in order to track ownership and make sure that only eligible investors can hold tokens, in order to comply with securities laws.

Global regulatory bodies now increasingly recognize these tokens as securities, requiring enforcement of compliance with existing securities laws, including stringent Know Your Customer (KYC) and Anti-Money Laundering (AML) regulations.

The open-source ERC3643 token standard and its T-REX implementation were designed to address this need to support compliant issuance and management of permissioned tokens, that are suitable for

[1] In this document, we will refer to:

- **Utility tokens**: are cryptocurrency tokens that merely grant token holders access or the right to participate on platform(s). When you think of Initial Coin Offerings (ICOs) utility tokens are the tokens offered to investors and grant the investor zero rights to the underlying issuers business.
- **Security tokens** (or **digital securities**/**tokenized securities**): are securities represented on a blockchain. These tokens grant investors rights akin to those of traditional securities, encompassing equity, debt, and more. They can represent any asset class, including small businesses and real estate. Issuers can conduct Security Token Offerings (STOs) to raise funds, however, security tokens are securities, required to comply with traditional securities laws.

tokenized securities, either on a peer-to-peer basis or through regulated trading platforms. These tokens are issued in full compliance with the rules specified by the investors (via on-chain identity) and the offerings based on issuers' guidelines. Furthermore, control mechanisms are baked into the tokens themselves.

Adopting a “Compliance by Design” approach, T-REX ensures that an investor cannot become a holder of any digital securities without fulfilling all compliance requirements. Furthermore, regulators can affirm the issuer's compliance by auditing the smart contracts that underpin the entire life cycle of the security token. This innovation offers a secure, transparent, and efficient environment for managing security tokens while enforcing on-chain compliance, heralding a new era in the financial securities market.

The management of compliant transactions through T-REX backed permissioned tokens will be based on 4 main pillars creating a **decentralized validator:**

- ONCHAINID[2], a blockchain based **identity management system**, allowing for the creation of a globally accessible identity for every stakeholder.
- A **set of validation certificates**, or verifiable credentials, (technically speaking, these certificates are the claims, described in the ERC-735 standard used by ONCHAINID, that will be described further in the document. For a better understanding, we will name these as certificates in this introductory section as it has more semantic sense.) emitted by trusted third parties and signed on-chain, each of them linked to a single ONCHAINID.
- An **Eligibility Verification System (EVS)** whose role is to act as a filter of all the transactions of tokenized securities and will check the validation certificates of the stakeholders. Essentially, the EVS will check that the receiver has the rights to receive the tokens following the specific offering rules and issuer requirements, for investors, applicable for this specific asset. The EVS will block the transaction if the receiver misses a mandatory certificate and will notify them about the reason for the failure. The onchain validator is implemented on the Identity Registry smart contract through the “isVerified” function.
- A set of **Compliance rules** (i.e. offering rules) ensuring that the rules of the offering are respected, e.g. the maximum of investors per country of distribution, the maximum of tokens held by a single investor, etc. These rules are not only linked to the identity of the receiver of a transaction but also to the global distribution of tokens at a certain time. The Compliance rules are implemented on the Modular Compliance smart contract through the “canTransfer” function.

These 4 key elements allow issuers to use a **decentralized Validator to control transfers and enforce compliance on the holders of the security token**. The Validator includes rules for the whole offering (e.g. managing the max number of holders allowed in a specific market, when such rules apply), and rules

[2] https://ONCHAINID.com/

for each investor (e.g. KYC or issuer-defined eligibility criteria) thanks to the identity management system.

Constraints for tokenized securities

While the rules governing utility tokens and their issuance remain relatively undefined or vague in most jurisdictions, Security Token Offerings (STOs) represent a different paradigm. STOs utilize blockchain technology as a registry, proof of ownership, and transfer infrastructure for securities, which are regulated instruments in every country. Consequently, STOs must comply with the relevant regulations in the countries where the security tokens are issued and distributed.

	Utility Token	**Security Token**
Purpose	Usage	Investment
Regulation	Non-existing or vague in most cases	Stringent as existing securities laws should be taken as reference
Life cycle	Simple	As complex as a security
Secondary Market	Nearly no constraints	As complex as a security

Figure 1 : comparison between utility and security token

A key distinction between Initial Coin Offerings (ICOs) and STOs resides in the token lifecycle. ICOs, dealing with utility tokens, yield tokens with a relatively straightforward lifecycle: once distributed across a decentralized network, their governance primarily stems from their token economics. However, for security tokens, the landscape is distinct. The issuer, or its appointed agent, generally remains liable for enforcing controls post-issuance and throughout the entire lifespan of the security token. Additionally, the issuer may need to execute corporate actions (like dividend/interest payments) or corporate events (such as calling for an AGM/EGM), necessitating ongoing interaction with (and some control over) their investors.

Two main types of control requirements are associated with the issuance, holding, and transfer of security tokens:

- **General Regulatory Controls:** These are regulations applicable to the security in question, independent of the token itself. For instance, obligations related to Anti-Money Laundering (AML) and Know Your Customer (KYC) protocols, such as investor identification, proof of identity verification, and blacklist checks, fall under this category.

- **Specific Security Controls:** Certain controls might be tied specifically to the security being issued. These could include restrictions related to the investor's type, location, or investment limit within a specific period. Such controls may arise from the regulatory environment the issuer operates within or be linked to eligibility criteria defined by the issuer for commercial reasons (e.g., limiting access to a specific share class with distinct fee characteristics to investors from a particular country).

To meet these diverse control requirements, a high level of reusability and flexibility is crucial in token design. This inspired us to develop the ERC3643 "T-REX" standard. The T-REX standard provides a suite of generic tools that aid token issuers in implementing and managing necessary controls and permissions for security tokens. This is achieved through a flexible decentralized validation system (EVS + Compliance contract), enabling relevant parties to establish necessary rules for approving the holding and transferring of their tokens.

Decentralized Validation System

Overview

Transfers of ERC-20 tokens typically proceed in a specific manner on any Ethereum Virtual Machine (EVM) blockchain, adhering to the standard ERC-20 implementation:

Figure 2 : illustration of an ERC-20 transaction

Transactions occur directly between two peers without restrictions or controls. The transactional freedom is comprehensive and pseudonymous, with AML/KYC checks primarily conducted when cryptocurrencies convert into fiat currencies or vice versa. However, various methods allow circumventing these checks, such as unregulated exchanges or direct peer-to-peer exchanges.

In contrast, a transaction involving T-REX compliant ERC-20 permissioned tokens follows a more controlled process:

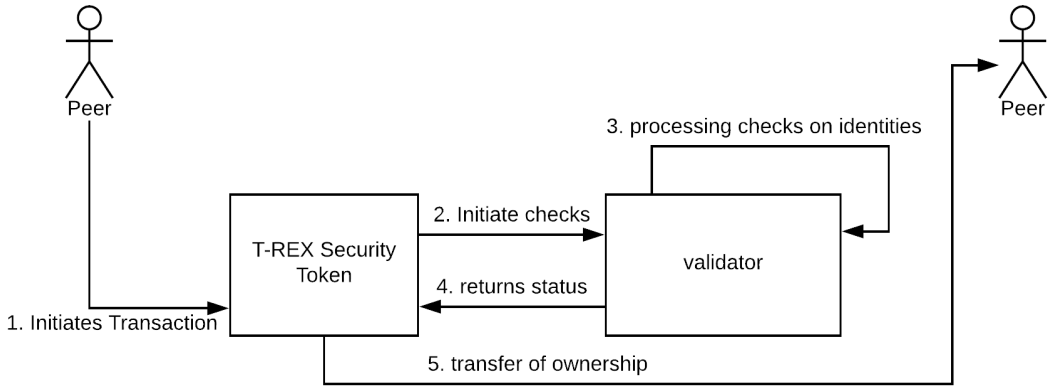

Figure 3 : illustration of an ERC3643 "T-REX" permissioned token transaction

1. **Transaction Initiation:** The token holder initiates the transaction via the security token smart contract. This action differs from a standard ERC-20 token, as the smart contract's transaction function has been modified.
2. **Validator Engagement:** The smart contract's transfer function calls upon the validator (comprising the compliance contract, identity registry, identity registry storage, trusted claim issuer registry, and trusted claim topics registry) to initiate checks on the receiver's ONCHAINID.
3. **Compliance and Eligibility Checks:** The validator ensures that the receiver's ONCHAINID holds the necessary claims, issued by a trusted third party listed in the trusted issuers registry. It also verifies that the proposed transfer complies with the rules set forth in the compliance smart contract.
4. **Evaluation of Transfer:** If the receiver's ONCHAINID possesses the requisite claims (personal data validated by trusted third parties such as KYC, AML, sovereign identity, etc.), and the transfer does not violate any compliance rules, the transfer of tokens is allowed to proceed. However, if the ONCHAINID lacks necessary claims, or the transfer breaches any compliance rule, the transfer is rejected.
5. **Transfer Execution or Rejection:** If the checks are successful, the transfer of tokens is executed. In case of a rejection, an error message is generated, explaining the necessary steps to acquire the missing claims or the reason for the transfer's non-compliance.

This process embodies the ethos of T-REX: ensuring compliant and secure transactions for all parties involved in the token exchange.

ERC3643 Permissioned tokens

We strongly believe that permissioned tokens, such as ERC3643, are the most suitable for issuing security tokens. The reason for this lies in the inherent need for controlled transactional freedom when dealing with such financial instruments, as investors must meet certain eligibility criteria. These criteria can be regulatory in nature or stipulated by the issuer themselves.

The primary technical distinction between standard ERC-20 tokens and ERC3643 permissioned tokens lies in the conditional nature of the transfer function in ERC3643 tokens. A transaction can only be executed if a decentralized validator approves it, based on the specific governance criteria defined for the token.

Despite this modification to the transfer function, it's important to note that ERC3643 tokens maintain full compatibility with all ERC-20 based exchanges and tools due to their underlying structure. Integration into an existing ERC-20 compatible platform requires a minor modification: processing pre-checks before any transfer to verify the transaction's compliance status with the decentralized validator. If a transaction fails to meet compliance, the platform should provide clear feedback on why it's non-compliant and, where possible, guidance on achieving compliance. However, in certain cases, compliance may not be achievable, for example, when attempting a transfer to a resident of a country listed on the T-REX compliance blacklist.

In the realm of "security token protocols", most solutions promoted so far are indeed permissioned tokens. They feature a modified transfer function that requires approval from an external validator service to control token transfers. T-REX goes a step further with its implementation, incorporating a fully on-chain identity management system. This allows issuers to have direct control over the transfer of ownership, enhancing the security and compliance of the system.

Onchain identities management

With security tokens subject to rigorous governance, adhering to all relevant regulations, particularly Know Your Customer (KYC) rules, is paramount. As such, we posit that effective identity management is crucial to ensuring this compliance on the blockchain.

Given that security token ownership is recorded on the blockchain, it's essential to track token ownership and inhibit illicit transactions directly on-chain. To this end, we propose linking wallet addresses to identities via an ONCHAINID contract on the blockchain itself. However, to maintain privacy and comply with personal data regulations, we suggest storing only validation certificates (claims) issued by

trusted third parties (KYC providers, government entities, lawyers, etc.) on-chain, rather than personal data. These trusted parties validate the data off-chain, and their certificates are utilized by the decentralized validator to determine whether parties can hold or transfer a specific security token.

Connecting an investor's wallet to an ONCHAINID can bring substantial benefits to stakeholders in the burgeoning security tokens market. For instance, should an investor lose access to their wallet, a token issuer can use the linked ONCHAINID to recover the investor's tokens. This process involves verifying the investor's personal data provided during recovery against the off-chain data associated with the ONCHAINID contract tied to the lost wallet. Once the investor's identity is confirmed, the issuer's agent can initiate the recovery function. This action forces a transfer of tokens from the lost wallet to a newly provided wallet, while also updating the identity registry and the ONCHAINID contract.

Furthermore, ONCHAINIDs and the stored certificates can be reused for passing KYC for other security tokens or even in scenarios beyond investment (e.g., account opening at an exchange, identification with compatible web services). In the realm of the Internet of Value, ONCHAINIDs could become as ubiquitous as Google and Facebook accounts in the Internet of Information, with the added benefit of being genuinely owned and controlled by their users.

T-REX infrastructure

Overview

T-REX aims to deliver a comprehensive toolset for issuing, managing, and transferring security tokens on EVM blockchains, leveraging the power of the ERC3643 permissioned token standard. The subsequent sections will delve into T-REX's technical intricacies, explaining the functions and capabilities of the various smart contracts integral to its operation. Furthermore, T-REX is designed with extensibility in mind, allowing for easy integration of additional smart contracts to manage corporate actions, taxes, and other related aspects.

Based on standards

Compatibility with Token standards

ERC-20

ERC-20 tokens[3] are recognized as an industry standard, embraced by all players in the blockchain sector. These tokens are fungible, often non-permissioned, and facilitate effortless peer-to-peer transfers on EVM blockchains.

[3] https://github.com/OpenZeppelin/openzeppelin-solidity/tree/master/contracts/token/ERC20

The ERC-20 smart contract outlines and implements the fundamental attributes of the token, including its name, symbol, total supply, and the number of decimals for display purposes. Essentially, the ERC-20 smart contract equips the token with all the requisite functionalities to meet the standards of a conventional token.

Identity standards on the Blockchain

Incorporating robust KYC and AML procedures is non-negotiable for any project seeking to adhere to stringent governance standards and regulatory compliance. This is particularly relevant for Security Token Offerings (STOs), which are mandated by regulations to enforce strict KYC and AML protocols. We believe that the most effective way to implement these checks within a blockchain infrastructure involves linking them to a universally accepted identity model.

To this end, we have adopted ONCHAINID, which is built on the ERC-734[4] and ERC-735[5] standards. These standards provide a widely accepted model for creating, maintaining, and populating identities on the blockchain. By leveraging these standards, we've built functionalities into our smart contracts that permit interactions only from well-identified, reputable entities, be they individuals or organizations. The claims attached to an identity contract foster a web of trust among token issuers, third-party claim issuers, and the identity contract holder, thereby streamlining identity management.

To add a claim to their ONCHAINID, an identity holder must first request it from a relevant trusted third party (a trusted claim issuer for the specific claim topic, as listed in the trusted issuers registry smart contract). Upon verifying the requestor's eligibility for the claim, the claim issuer signs a message containing: a) the requestor's ONCHAINID address, b) the claim topic, c) optional data (e.g., hashes to off-chain data references stored by trusted claim issuers), and d) a valid signature of the claim information by a key listed on the Claim Issuer smart contract. The identity owner (the claim holder) then stores this claim in their identity contract. Alternatively, the claim issuer can add the claim themselves, provided the identity owner has given approval.

In the T-REX framework, it's this synergy of identities and claims that enables on-chain validation and verifies an investor's eligibility to hold a specific security token. Each time an investor is slated to receive a position in a specific security token, T-REX verifies whether their ONCHAINID contains the necessary claim(s) to hold that token.

[4] https://github.com/ethereum/EIPs/issues/734
[5] https://github.com/ethereum/EIPs/issues/735

Proxy standards

ERC-1822

ERC-1822[6] establishes a universal standard for proxy contracts, guaranteeing compatibility with all other contracts. It achieves this compatibility by storing the Logic Contract's address in a distinct storage location within the proxy contract. A compatibility check also allows for successful upgrades, which can be performed indefinitely or as defined by custom logic. Furthermore, the standard provides a method to choose from multiple constructors without inhibiting bytecode verification.

In the context of the T-REX protocol, we adopt an adapted version of the Universal Upgradeable Proxy Standard (UUPS) proposed by ERC-1822. However, we do not directly store the implementation contract's address in the proxy storage as per the ERC-1822 standard. Instead, the T-REX proxy contract stores the address of an external contract, known as the Implementation Authority.

Beacon Proxy[7]

The Beacon Proxy Standard introduces another approach to managing contract upgrades. In its structure, a beacon proxy points to a beacon contract which stores the address of the current implementation. This beacon contract can update its implementation address, thereby upgrading all beacon proxies that reference it.

In the T-REX protocol, we harness the structure of the Beacon proxy, but implement it like an ERC-1822 contract. Our proxy contract references an external contract, the Implementation Authority, which contains the addresses of all T-REX contract implementations. This Implementation Authority address fits into the storage slot that the ERC-1822 standard reserves for implementation.

The benefits of the T-REX protocol's method include centralized versioning management for all T-REX tokens deployed using the same Implementation Authority address, making it an ideal choice for simultaneous token upgrades. Token issuers can change the address of their tokens' Implementation Authority contract, giving them the power to manage versioning independently or delegate it to any trusted third party. The T-REX factory contract sets the Implementation Authority address during token deployment by default.

[6] https://eips.ethereum.org/EIPS/eip-1822
[7] https://docs.openzeppelin.com/contracts/3.x/api/proxy#beacon

T-REX Components (Smart contracts library)

Below is the illustration of T-REX components with global interactions:

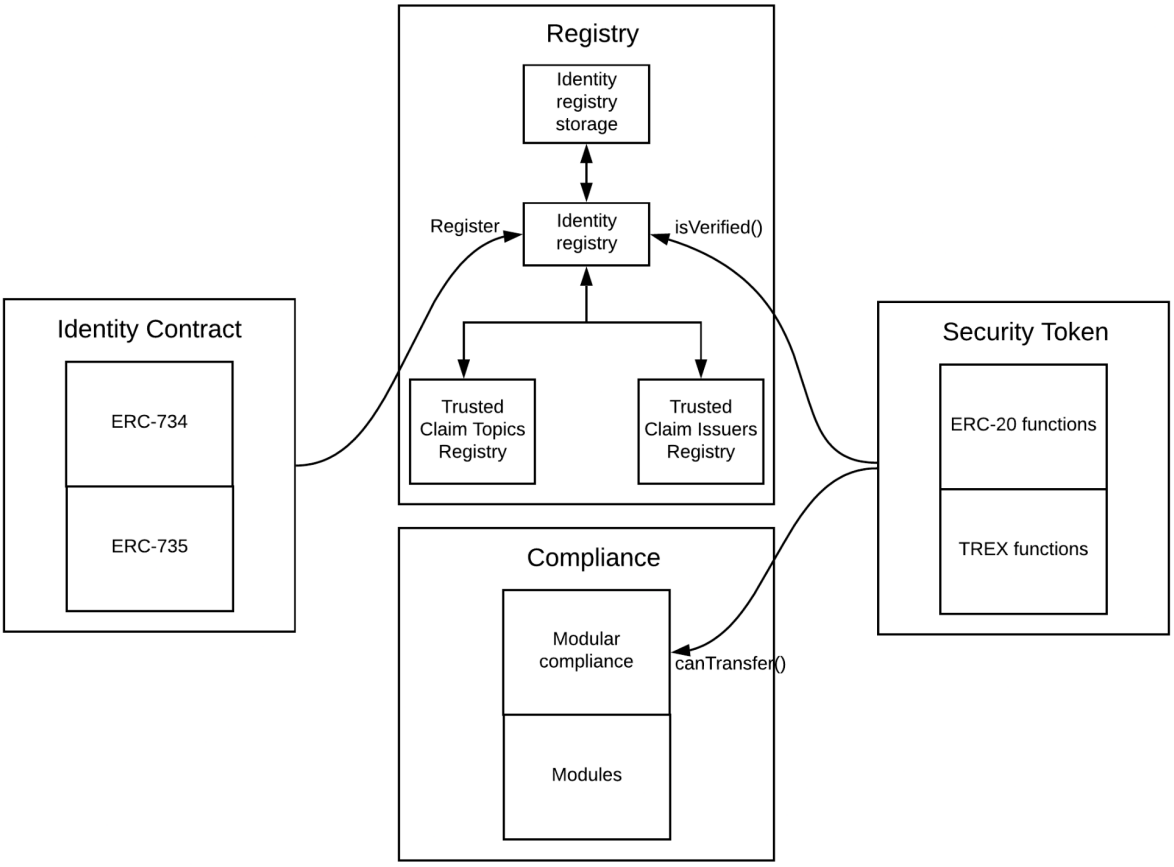

Figure 4 : illustration of T-REX components with global interactions

ONCHAINID

An ONCHAINID contract is a smart contract deployed by a user to interact with the security token, or for any other application where an onchain identity may be relevant. Based on the ERC-734 and ERC-735 standards, this contract stores keys and claims related to a specific identity, and encompasses all essential functions for managing these elements.

The ONCHAINID contract is not tied to a specific token, meaning each user needs to deploy it only once. Afterward, it can be used in any context where an onchain identity may be beneficial. A comprehensive description of the functions is available in the ONCHAINID documentation[8].

Moreover, an ONCHAINID contract is also deployed and associated with the token smart contract to represent the identity of the financial asset itself. The claims provide information about the asset and all pertinent corporate actions and details regarding the security. Essentially, it serves as an onchain "golden copy" of the asset, and can be augmented with any relevant claim throughout the token's life cycle.

Identity Registry

The Identity Registry smart contract serves as the execution hub for the Eligibility Verification System (EVS). It establishes connections with the Trusted Issuers Registry and Claim Topics Registry to stipulate identity requirements and executes the "isVerified" function. This function compares the EVS requirements with the identities of investors to ascertain their eligibility status.

The Identity Registry contract is linked to the Identity Registry Storage contract, which houses a dynamic "whitelist" of identities. Utilizing the Identity Registry Storage, the Identity Registry can retrieve the association between a wallet address, an ONCHAINID, and a country code that denotes the investor's country of residence. This country code is set in compliance with the ISO-3166 standard[9].

The management of the Identity Registry lies with the issuer's agent(s), meaning only the issuer's agent(s) can execute functions to add or remove identities from the Identity Registry Storage. It's worth noting that the agent role on the Identity Registry is assigned by the issuer, who can designate themselves as the agent if they prefer to maintain complete control.

Each security token has a specific Identity Registry, as each one is associated with a Claim Topics Registry and a Trusted Issuers Registry smart contracts. These contracts dictate the eligibility rules of the token and are unique to a token. A comprehensive description of the functions is available in the T-REX contracts documentation[10].

Identity Registry Storage

The Identity Registry Storage contract functions as a repository for a mapping table, associating wallet addresses with the corresponding ONCHAINID addresses of investors. This correlation enables the Identity Registry to extract the ONCHAINID linked to a specific wallet, thereby retrieving the claims

[8] https://docs.onchainid.com/
[9] https://www.iso.org/iso-3166-country-codes.html
[10] https://docs.tokeny.com/docs/identity-registry

associated with that particular ONCHAINID address. The roster of wallets and identities contained in this mapping encompasses all individuals whose data the issuer possesses and might need to utilize.

At its most basic level, this list should incorporate investors who have undergone the necessary Know Your Customer (KYC) and eligibility evaluations, and are consequently authorized to hold the issuer's security tokens. Moreover, it can also include potential future investors.

The Identity Registry Storage can be tied to one or multiple Identity Registry contracts. Its primary objective is to segregate the Identity Registry's functions and specifications (which determine eligibility rules specific to a token) from its storage. This arrangement permits the maintenance of a separate Eligibility Verification System (EVS) for each token while sharing the list of investors vetted by the isVerified() function, implemented in the Identity Registries. This function checks the eligibility of the receiver in a transfer transaction. A thorough description of the functions is available in the T-REX contracts documentation[11].

Trusted Issuers Registry

The Trusted Issuers Registry smart contract serves as a storage system for contract addresses (ONCHAINIDs) of all trusted claim issuers specific to a given security token. Trusted claim issuers, along with the claim topics they are authorized for, reside within the Trusted Issuers Registry, with each issuer possessing their unique set of responsibilities concerning the claims they are permitted to issue.

To qualify for holding the token, the ONCHAINID of token owners (the investors) must possess claims endorsed by the claim issuers housed in this smart contract. The token issuer retains the ownership of this contract, providing them with the ability to manage this registry in accordance with their requirements. A comprehensive description of the functions is available in the T-REX contracts documentation[12].

Claim Topics Registry

The Claim Topics Registry smart contract houses all the claim topics necessary for holding the security token. The ONCHAINID of token owners (the investors) must possess claims of the topics stored within this smart contract, which must be issued by the corresponding trusted issuers as cataloged in the Trusted Issuers Registry.

Ownership of this contract is vested in the token issuer, providing them with the autonomy to manage this registry in line with their specific needs. An exhaustive explanation of the functions is made available in the T-REX contracts documentation[13].

[11] https://docs.tokeny.com/docs/identity-registry-storage
[12] https://docs.tokeny.com/docs/trusted-issuers-registry
[13] https://docs.tokeny.com/docs/claim-topics-registry

Permissioned Token

T-REX security tokens are built on the fundamental structure of the established ERC-20 standard, but are enriched with additional functions to ensure full regulatory compliance in security token transactions. The transfer and transferFrom functions are designed conditionally, executing a transfer solely upon the approval of the decentralized validator (EVS + Compliance). It means that transfers can only be triggered when both investor rules (via EVS) and offering or additional rules (via Compliance) are met.

This ensures that the permissioned tokens can only be transferred to verified parties, thereby preventing the tokens from landing in the wallets or ONCHAINIDs of ineligible or unauthorized investors. The T-REX standard also features a robust mechanism for the recovery of security tokens, should an investor lose access to their wallet's private key. A historical record of all recovered tokens is meticulously maintained on the blockchain for the sake of transparency.

Beyond these features, T-REX tokens implement a plethora of additional functions. These empower the token owner or their designated agent(s) with the capability to manage aspects such as supply, transfer rules, lockups, and any other facets integral to managing a security at their discretion. For a detailed explanation of these functions, please refer to the T-REX contracts documentation.[14].

Modular Compliance

The Compliance smart contract is a dynamic tool used to establish the rules of the token offering. These rules, once set, are meticulously adhered to throughout the entire lifecycle of the token. For instance, the compliance contract can define the maximum number of investors per country, the maximum quantity of tokens per investor, and the countries where the token can circulate. The latter is achieved by using the country code corresponding to each investor in the Identity Registry.

Designed with a modular approach, the compliance smart contract grants Token Issuers the flexibility to add or remove compliance modules. Each of these modules corresponds to a distinct and independent compliance rule. This flexibility enables the issuer to shape compliance rules based on the specific requirements of the token transfers.

The compliance contract is activated at every transaction by the Token. It then evaluates whether the transaction is in compliance with the rules of the offering. It returns a TRUE if the transaction adheres to the rules and a FALSE if it does not. For an in-depth description of the functions, please refer to the T-REX contracts documentation[15].

[14] https://docs.tokeny.com/docs/token-smart-contract
[15] https://docs.tokeny.com/docs/compliance

Implementation Authority

The Implementation Authority contract, as previously discussed in the proxy section, is a pivotal contract that oversees the versioning of T-REX contracts utilized by proxies. It operates in two distinct capacities:

- Main Implementation Authority Contract: This contract is utilized by the T-REX factory to deploy all T-REX tokens. The owner of this contract can add new versions of the T-REX contracts and decide to upgrade all proxies linked to the implementation authority to a different version. Furthermore, it enables token owners to deploy auxiliary Implementation Authority contracts and modify the contract their proxies use for versioning.
- Auxiliary Implementation Authority Contracts: These are deployed when a token issuer wishes to change the reference Implementation Authority contract for their proxies. Unlike the main contract, auxiliary contracts cannot add new versions; they can only fetch versions from the main contract, which serves as a constant reference. However, they can opt to run a different version at any given time compared to the main contract. The owners of auxiliary contracts have the flexibility to revert to the main contract whenever necessary.

For a comprehensive understanding of the functions, please refer to the T-REX contracts documentation[16].

Factory

The T-REX Factory Smart Contract embodies a robust utility, enabling the simultaneous deployment and configuration of all contracts within the T-REX suite in a single blockchain transaction. This transaction is characterized by its flexibility, accommodating numerous parameters that align with the unique requirements of the token issuer.

The T-REX Factory references the main Implementation Authority contract, as previously discussed. The owner of the T-REX factory retains the privilege to update this reference point to a different contract as needed.

All contracts deployed by the factory are in the form of proxies, pointing to the main Implementation Authority. These proxies are deployed utilizing the CREATE2 opcode. This allows for the deployment of a token at the same smart contract address across multiple EVM blockchains, provided the factory itself is deployed at an identical smart contract address on the different blockchains.

To prevent misuse, the T-REX deployment function is safeguarded by an “onlyOwner” modifier at the contract level. This is crucial to avoid scenarios where someone could deploy a token with the same

[16] https://docs.tokeny.com/docs/implementation-authority

address on a different blockchain without being the same owner in control across the blockchains, potentially leading to scams and confusion.

The owner, who has the ability to call the T-REX deployment function, can either be a wallet (though not recommended) or an external smart contract. The latter manages deployment roles and ensures the same token address cannot be deployed by different users on different blockchains. This can be achieved by utilizing oracles or simply incorporating the T-REX owner's wallet as part of the salt used by the CREATE2 opcode to generate deterministic addresses.

For an exhaustive understanding of the functions, please refer to the T-REX contracts documentation[17].

Additional smart contracts

In addition to the core T-REX suite, numerous supplementary smart contracts can be incorporated to enhance the functionality of security tokens and cater to the specific requirements of issuers and investors. By incorporating these features, the T-REX ecosystem is able to seamlessly integrate traditional securities market functions with the inherent benefits of blockchain technology.

- **Investor Rights and Opportunities:** Investor rights, such as voting, dividends, and announcements, can be easily implemented by the issuer or tokenization platform. This allows for the creation of a more comprehensive and adaptable security token ecosystem. Other potential investor rights that could be integrated into the protocol include:
    - Participation in corporate governance decisions
    - Real-time access to company financial information
    - Transparent communication channels between issuers and investors
- **Harnessing the Power of Blockchain:** Tokenized securities can leverage the unique capabilities of blockchain technology to improve market efficiency and user experience. By implementing additional smart contracts, the T-REX protocol can offer:
    - Enhanced transparency for all market participants
    - Elimination of multi-stage reconciliations
    - 24/7 processing and global reach
    - Improved security and immutability of records
- **Tax Compliance and Automation :** The T-REX protocol can also incorporate smart contracts that address tax compliance, automating the calculation and distribution of taxes on token transactions. This feature would greatly streamline the process and ensure regulatory compliance for both issuers and investors.

In summary, the T-REX protocol can be expanded to include a wide array of additional smart contracts, enriching investor rights, and harnessing the full potential of blockchain technology. By doing so, it sets

[17] https://docs.tokeny.com/docs/factory

the stage for a new era in the securities market, characterized by unparalleled efficiency, transparency, and global accessibility.

Stakeholders

Overview

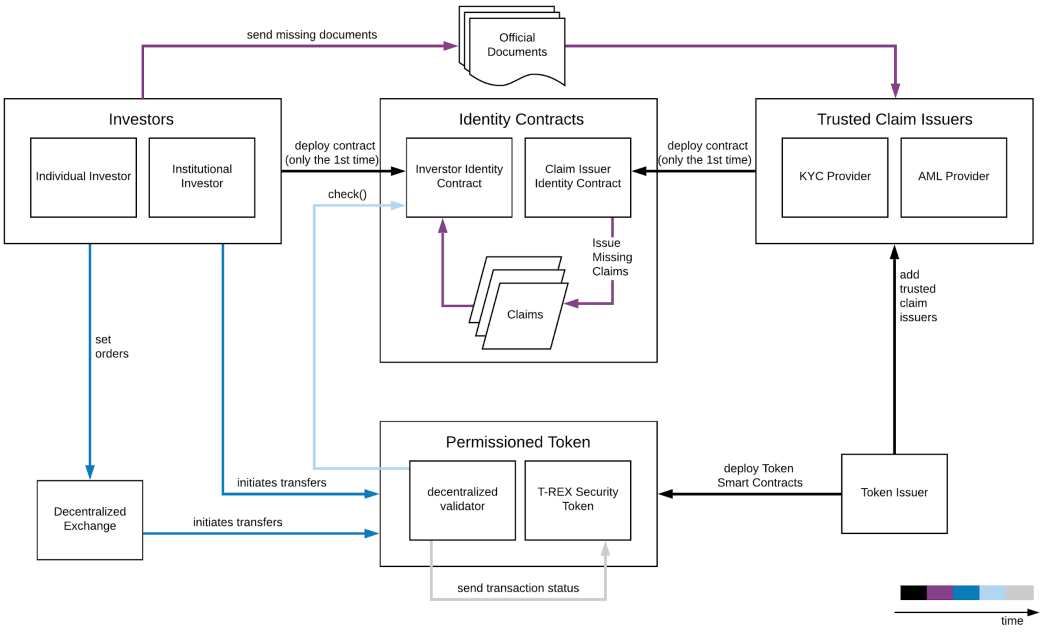

Figure 5 : Stakeholders of the ecosystem and their interactions

Issuer

As the primary stakeholder in the T-REX protocol, the issuer – the entity deploying the security token – assumes a vital and multifaceted role. Tasked with overseeing and managing the comprehensive suite of smart contracts, the issuer's responsibilities extend far beyond the initial token deployment.

- **Determining Claim Topics and Trusted Issuers:** The issuer's first responsibility is to establish the claim topics required by their investors and identify the trusted claim issuers who can verify those claims. These determinations are influenced by several factors including the jurisdiction of issuance, the intended distribution countries, and the unique attributes of the security token itself.
- **Engaging with Issuance Platforms:** Issuance platforms can provide invaluable assistance to issuers, helping them deploy and manage the necessary smart contracts that underpin their token.

Additionally, these platforms can facilitate investor engagement, assisting them in creating their ONCHAINID or connecting with an existing ONCHAINID, as required.

- **Administering the Suite of Smart Contracts:** Upon the deployment of a token, the issuer assumes ownership of the suite of smart contracts at the blockchain level. This places them in a pivotal role, responsible for administering the token and adding agents. While the issuer can execute these tasks independently, they may also utilize interfaces provided by a tokenization platform for streamlined management.

In summary, the issuer's role within the T-REX framework is comprehensive and multifaceted, extending from the initial determination of claim topics and trusted issuers to the ongoing administration of the suite of smart contracts. Their role is central to the successful deployment and management of security tokens within the T-REX ecosystem.

Investors

Investors form the backbone of the T-REX ecosystem, with their actions and interactions underpinning the operational integrity of the system. As participants, they maintain control over their unique ONCHAINID, managing access permissions for claim issuers and checkers as required.

- **Control Over ONCHAINID:** Investors create and exercise control over their ONCHAINID. This allows them to dictate the terms of access for claim issuers and checkers, entities that may require access to private, off-chain data related to claims. This control ensures the investor's privacy and security in their interactions within the T-REX ecosystem.
- **Data Protection and Privacy:** The T-REX protocol is designed to safeguard the sensitive data of investors and all other stakeholders. Direct access to this data on the blockchain is not permitted. Instead, only assessments from trusted claim issuers are linked to an investor's identity contract. These assessments are verified by the Identity Registry, which approves or denies transactions initiated on the token smart contract, in compliance with data protection laws.
- **Simplifying Identity Verifications:** Through their ONCHAINID, investors have the potential to streamline their participation in multiple token issuances. If the claims stored on their ONCHAINID meet the KYC and eligibility requirements of subsequent token issuers, investors may not need to undergo identity verification for each issuance. Instead, they can provide direct access to the necessary claim checks. If the security token issuer recognizes the claim issuer as a trusted entity (for instance, acknowledging the legitimacy of the Estonian digital identity), the validation process is further simplified.

In summary, investors play a critical role in the T-REX ecosystem, from managing their ONCHAINID to navigating the identity verification process. Their actions and decisions shape the operational dynamics of the system while ensuring their data remains protected and privacy intact.

Claim issuers

Claim issuers occupy a pivotal role within the T-REX ecosystem, authorized by token issuers (e.g., a designated third-party KYC platform tasked by the issuer to conduct KYC checks) and registered in the Trusted Issuers Registry. As a Trusted Claim Issuer, they have the authority to augment a specific investor's ONCHAINID.

Claim issuers can maintain multiple signer keys (at least one) in their own ONCHAINID. This model fosters true interoperability among multiple KYC/AML providers. By employing a standard protocol to verify claims, it insulates Issuers from the specificities of individual AML/KYC providers. The Claim Issuers, therefore, serve as an essential link in the T-REX ecosystem, facilitating the addition and verification of claims while promoting a unified and interoperable environment.

Distributors, Exchanges and DeFi

In the T-REX ecosystem, exchanges also need to manage identities and hold claims to qualify as eligible token holders and accept T-REX token deposits. The integration of identity management within exchanges is a critical aspect that hinges significantly on the exchange's internal transfer mechanism. Depending on the specific transfer mechanism in place, the integration method of the identity protocol will vary. This foundational understanding will be further detailed and explored in the following subsections:

Direct P2P Trades

In the T-REX ecosystem, individual investors have the capability to directly trade tokens among themselves. However, the compliant service of the token restricts any transfer of tokens to unauthorized addresses. For a successful transfer, several conditions must be met:

1. The buyer must have a valid identity in accordance with the security token standards, and this identity must be registered in the identity registry.
2. The buyer must also satisfy the claim requirements of the token.
3. The token should not be within its lockup period.
4. Lastly, the token transfers must adhere to the compliance rules set forth in the compliance contract.

The T-REX repository provides an implementation of a Direct P2P Bilateral Transaction Manager, known as the Delivery-versus-Delivery (DVD) Transfer Manager. This contract ensures secure management of transfers, where a transfer can only proceed if the counter-transfer is successful between eligible investors, following an off-chain agreement about the transaction details. The sequence diagram below illustrates the operation of a DVD contract, including potential fees collection and distribution.

.

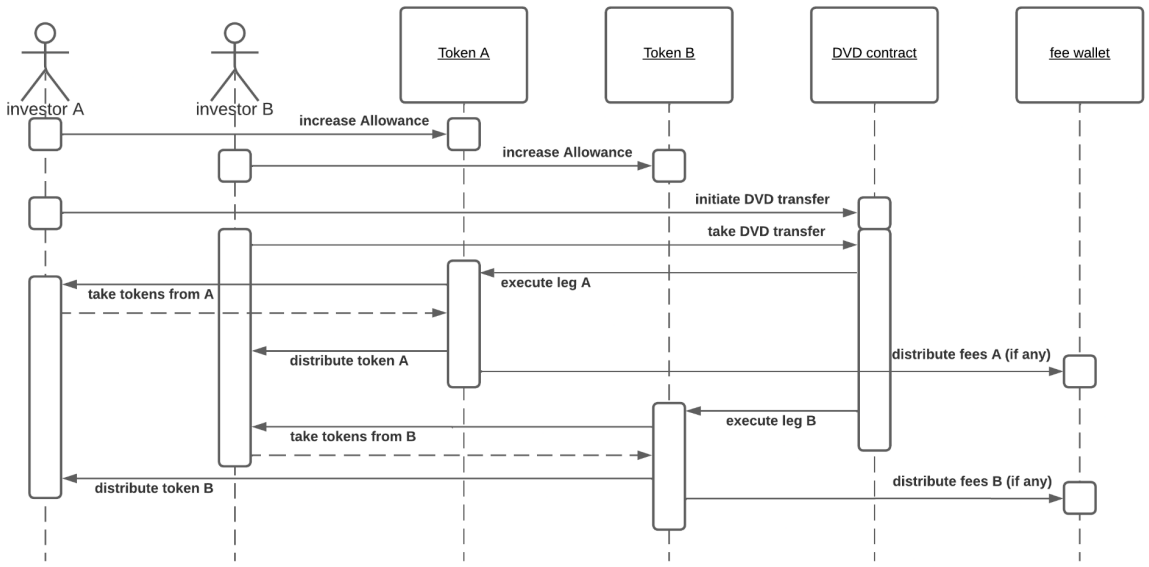

Figure 6 : sequence diagram of a DVD transfer

The steps for executing a DVD transfer are as follows:

1. Investors A and B agree on a DVD transfer, with Investor A needing to know Investor B's wallet address.
2. Investor A authorizes the DVD contract for Token A, the token they will send to Investor B in exchange for Token B. The granted allowance must be at least equal to the amount of Token A involved in the DVD transfer.
3. Similarly, Investor B authorizes the DVD contract for Token B, the token they will send to Investor A in exchange for Token A. Again, the granted allowance must be at least equal to the amount of Token B involved in the DVD transfer.
4. Investor A initiates the DVD transfer by calling the relevant function with the previously agreed parameters.
5. Investor B verifies the parameters set by Investor A against their preliminary agreement. If everything aligns, Investor B triggers the DVD transfer on the transfer ID created by Investor A by calling the relevant function.
6. The DVD smart contract completes the token distribution as per the DVD contract parameters in a single transaction. This ensures that if either investor fails to provide the promised tokens, the entire transaction is reversed, preventing any loss of tokens.

Decentralized Exchanges with off chain order book

Certain exchanges, such as those based on protocols like IDEX[18] or dYdX[19], allow direct address management to facilitate distributed transfers. These Decentralized Security Token Exchanges (STEs) can interact with the T-REX identity management and transfer protocol by adhering to the following process:

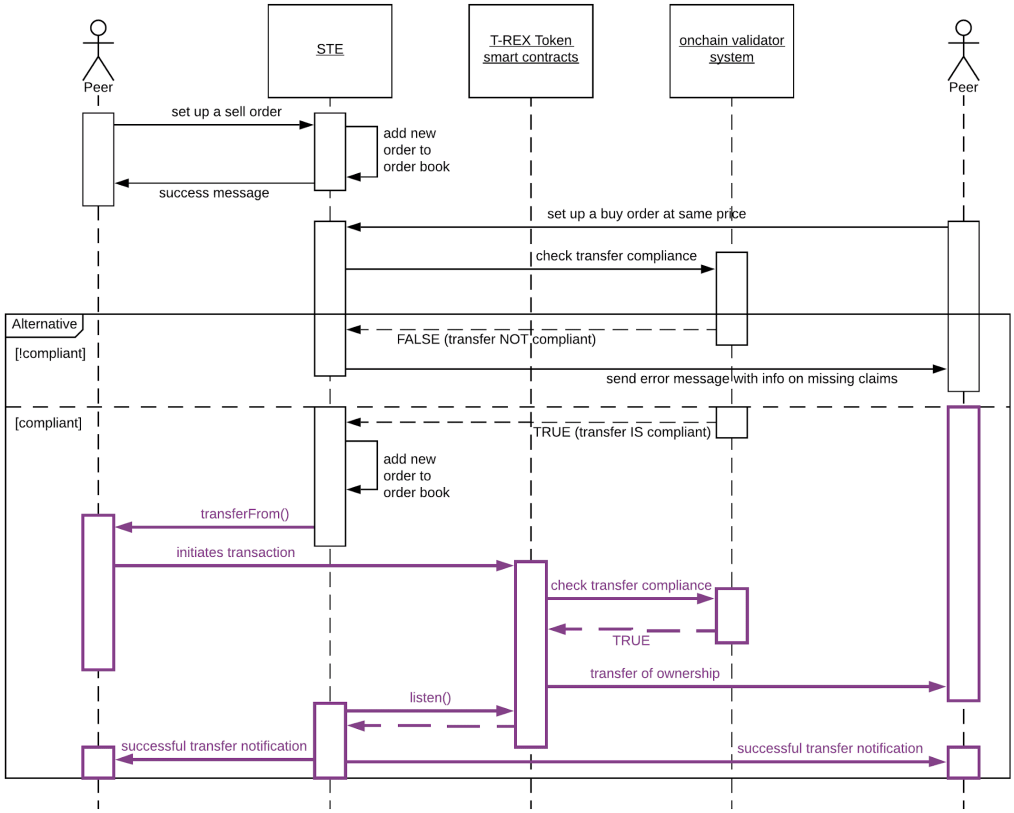

Figure 7 : sequence diagram of a Distributed Security Token Exchange transaction with T-REX tokens

1. A token holder logs into the STE and places a sell order.
2. The STE verifies the seller's balance and allowance to ensure they possess enough tokens to fulfill the order and confirms the exchange has permission to access the tokens for settlement.

[18] https://idex.io/
[19] https://dydx.exchange/

3. If the seller's balance and allowance meet the necessary criteria, the order is added to the STE's order book, and a success notification is sent to the order issuer. If the balance or allowance is insufficient, the order is rejected, and an error message is returned to the order issuer.
4. A prospective buyer places a buy order on the same token at a price equal to or higher than the existing sell order. The buyer must have sufficient balance and allowance on the token used for the purchase to add the order to the order book.
5. The STE checks the buyer's identity contract to confirm if they hold the necessary claims to receive the token being traded. It also verifies if the potential transfer aligns with the compliance rules stipulated in the compliance contract. This compliance check is performed using the same method and code used to verify compliance before executing a standard transfer.
6. The identity registry and the compliance module return the buyer's status. If the necessary claims are lacking, or if the potential transfer violates the compliance rules, the order is cancelled, and an error notification detailing the requirements for compliance is sent to the buyer. Depending on the reason for the failure, the buyer might be able to rectify the situation.
7. If the buyer has the necessary claims and the compliance rules are not violated, the buy order is added to the order book. The STE then initiates the transactions using the transferFrom() function. This function is invoked on the Token contract and checks the receiver's claims, similar to a standard P2P T-REX transaction. If all conditions are met (which should be the case, as the STE pre-checked the claims before adding the orders to the order book), the transfer is executed. While the process may vary slightly depending on the STE protocol used, this description represents the most typical scenario.
8. The STE monitors the blockchain, and once the transaction is successfully included in a block, it sends a success message to the transaction counterparties.

Through this process, the protocol ensures only trades validated by the decentralized validator are permitted on such exchanges. It also demonstrates how these exchanges can expand the system's reach by interacting with non-compliant peers. These peers receive customized error messages with step-by-step instructions on how to achieve compliance. The purple section in the diagram represents the on-chain processing, while the rest of the process is handled off-chain.

Decentralized Exchange - Automated Market Makers

Automated Market Makers (AMMs) represent the most prevalent type of Decentralized Exchanges at present. A prominent example of this standard is the Uniswap protocol. These exchanges facilitate instant liquidity, enabling token holders to "swap" their tokens for other tokens. The transaction quantity and value are dictated by the swap volume, the swap ratio versus the desired tokens in the available Liquidity Pools (LPs), and the token availability within these LPs. As the quantity of tokens to be swapped grows relative to the size of the Liquidity Pool, the swap rate becomes less favorable.

For T-REX tokens to be swappable on an AMM, the Liquidity Pool contracts, through which the T-REX tokens transit during the swap process, need to be eligible (associated with an eligible ONCHAINID holding the required claims) and compliant with the rules defined in the Compliance contract. Liquidity Pool addresses need to be added to the Identity Registry Storage, giving the Token Issuer and their agent(s) the authority to approve or reject an exchange pair. Once a pair receives approval, Liquidity providers can begin depositing tokens into the LP to start earning transaction fees on each swap.

The process for an investor aiming to buy Security Tokens would unfold as follows:

1. The investor navigates to the DEX platform and connects their wallet.
2. The investor chooses the two tokens to be swapped.
3. If one of these tokens is an ERC-3643 token, the DEX initiates the necessary functions to verify if the swap can occur on-chain (verifying eligibility with an isVerified() function call, ensuring compliance with a canTransfer() function call, checking the freezing status, and confirming the token is not paused).
4. If any of these checks fail, the DEX should display the reason and guide the user on how to rectify the issue (if feasible).
5. If all checks are successful, the DEX should permit the user to sign the swap transaction and broadcast it to the blockchain.
6. Once confirmed on-chain, the swap proceeds and tokens are distributed accordingly.

Centralized Exchange (CEX) - Investor Owned Wallet

In the context of centralized exchanges that utilize one wallet per investor, with the private keys held by the CEX, the deposit process requires a special approach. Wallets must be linked to the ONCHAINID of the exchange, enabling the tagging of wallets as exchange wallets on the compliance contract. This arrangement facilitates user-specific tracking of deposits by verifying the ONCHAINID of the depositor during a transfer from a wallet tied to an investor ONCHAINID to a CEX ONCHAINID.

For example, this procedure can set monthly limits on deposits and withdrawals on CEX per investor. Prior to the deposit taking place, the investor's CEX wallet must be registered in the Token's Identity Registry and linked to the ONCHAINID of the CEX to facilitate transfers.

Here's the sequence of events for depositing tokens to a CEX:

1. An investor links their CEX wallet to the ONCHAINID of the exchange. This step is crucial for ensuring that the exchange can track the wallet as part of its operations and be able to enforce any necessary compliance rules.
2. The linked wallet is then registered in the Identity Registry of the Token by a token agent. This registration ensures that the wallet is recognized by the token's system and can interact with it.

3. Once the registration is complete, the investor can initiate the deposit. The system checks the ONCHAINID of the depositor to validate the transfer from the investor's wallet to the CEX ONCHAINID.
4. Depending on the compliance rules set on the contract, certain restrictions may be enforced. For instance, monthly limits on deposits and withdrawals per investor can be implemented.
5. The transfer of tokens is then completed, and the investor's tokens are deposited into the CEX.

However, it's important to note that the activities within the CEX cannot be tracked and validated by the Eligibility Verification System (EVS) or by compliance measures. Therefore, the CEX has the responsibility to ensure the compliance and eligibility of investors.

In order to provide a complete cap table, the CEX should also supply data about token holders to the token issuer. This information exchange is crucial for maintaining transparency and ensuring that all parties involved have a comprehensive understanding of the token distribution.
In the context of centralized exchanges that utilize one wallet per investor, with the private keys held by the CEX, the deposit process requires a special approach. Wallets must be linked to the ONCHAINID of the exchange, enabling the tagging of wallets as exchange wallets on the compliance contract. This arrangement facilitates user-specific tracking of deposits by verifying the ONCHAINID of the depositor during a transfer from a wallet tied to an investor ONCHAINID to a CEX ONCHAINID.

For example, this procedure can set monthly limits on deposits and withdrawals on CEX per investor. Prior to the deposit taking place, the investor's CEX wallet must be registered in the Token's Identity Registry and linked to the ONCHAINID of the CEX to facilitate transfers.

Here's the sequence of events for depositing tokens to a CEX:

6. An investor links their CEX wallet to the ONCHAINID of the exchange. This step is crucial for ensuring that the exchange can track the wallet as part of its operations and be able to enforce any necessary compliance rules.
7. The linked wallet is then registered in the Identity Registry of the Token by a token agent. This registration ensures that the wallet is recognized by the token's system and can interact with it.
8. Once the registration is complete, the investor can initiate the deposit. The system checks the ONCHAINID of the depositor to validate the transfer from the investor's wallet to the CEX ONCHAINID.
9. Depending on the compliance rules set on the contract, certain restrictions may be enforced. For instance, monthly limits on deposits and withdrawals per investor can be implemented.
10. The transfer of tokens is then completed, and the investor's tokens are deposited into the CEX.

However, it's important to note that the activities within the CEX cannot be tracked and validated by the Eligibility Verification System (EVS) or by compliance measures. Therefore, the CEX has the responsibility to ensure the compliance and eligibility of investors.

In order to provide a complete cap table, the CEX should also supply data about token holders to the token issuer. This information exchange is crucial for maintaining transparency and ensuring that all parties involved have a comprehensive understanding of the token distribution.

Centralized Exchange - Pooled Wallets

Unlike the case with investor-owned wallets, some centralized exchanges operate on a model where investors deposit their tokens into a few pooled wallets managed by the exchange. In this model, the CEX must have a mechanism to reconcile deposits with the corresponding user accounts, as the sender address is key to associating deposits with the correct users.

Here is the sequence of events for depositing tokens to a CEX using pooled wallets:

1. An investor initiates a deposit to one of the pooled wallets managed by the CEX. As these wallets are already registered in the Token's Identity Registry and linked to the ONCHAINID of the CEX, no additional linkage is necessary on the part of the investor.
2. The system checks the sender address of the deposit. This step is critical, as the sender address is used to reconcile the deposit with the correct user account on the CEX.
3. As with the investor-owned wallet model, compliance rules set on the contract can enforce certain restrictions. For instance, monthly deposit and withdrawal limits per investor can be implemented.
4. Once the sender address is verified and any compliance rules are satisfied, the transfer of tokens is completed, and the investor's tokens are deposited into the pooled CEX wallet.

Despite the difference in wallet management, the same challenge exists with regard to the cap table management for the token issuer. Since the on-chain cap table only recognizes the pooled wallets of the CEX, the internal distribution of tokens among individual investors within the CEX remains invisible on-chain. Consequently, it's the CEX's responsibility to ensure the compliance and eligibility of its users.

To maintain a complete and accurate cap table, the CEX must supply data about token ownership within the exchange to the token issuer. This exchange of information is vital for maintaining transparency and ensuring all parties involved have a comprehensive understanding of the token distribution.

T-REX processes

The ERC-20 token standard, with its transfer and transferFrom functions, provides the fundamental capability for token holders to transfer tokens to another address. Building upon this foundation, the T-REX standard introduces a permissioned token model on the EVM blockchains, allowing token transfers to take place only when approved by an on-chain Transfer Management Service. With appropriate configurations, T-REX facilitates compliant transfers, whether in Security Token Offerings (STOs), secondary trades, on exchanges (with a preference for decentralization), or peer-to-peer transactions, and it does so across various jurisdictions. Essentially, T-REX extends the functionality of ERC-20 tokens to support compliant security token issuance, management, and transfers.

T-REX modifies the standard ERC-20 methods, transfer() and transferFrom(), adding an extra layer of checks to determine if a transfer should be authorized. This verification process involves a review of the ONCHAINID contract of the recipient, ensuring that it possesses the necessary claims to receive or hold the security token. Additionally, the system calls upon the compliance contract to verify that the transfer aligns with the established rules for the token.

Within the T-REX ecosystem, there are four primary entities:

1. **Factory Deployer:** This is the party responsible for deploying the Library smart contracts and the Onchain T-REX Factory smart contract.
2. **Token Deployer:** This is the party responsible for deploying the security token's smart contracts.
3. **ABC Corporation:** This entity represents any company seeking to issue security tokens.
4. **Trusted Claim Issuer:** This party handles the issuance of claims to identity contracts (e.g., a third-party KYC provider). The Trusted Claim Issuer has the token issuer's confidence, and all token owners must possess claims signed by this entity.
5. **Investors:** These are the owners of security tokens who are authorized to trade tokens amongst themselves.

Security token deployment - standalone approach, no proxy

Role: Token Deployer

The deployment of a security token within the T-REX framework is a sequential process involving several interrelated smart contracts. Each contract performs a distinct role within the overall system. For a clear understanding of this process, we present the sequence diagram and the corresponding explanations that detail the steps involved in a standalone approach, i.e. without proxy contracts.

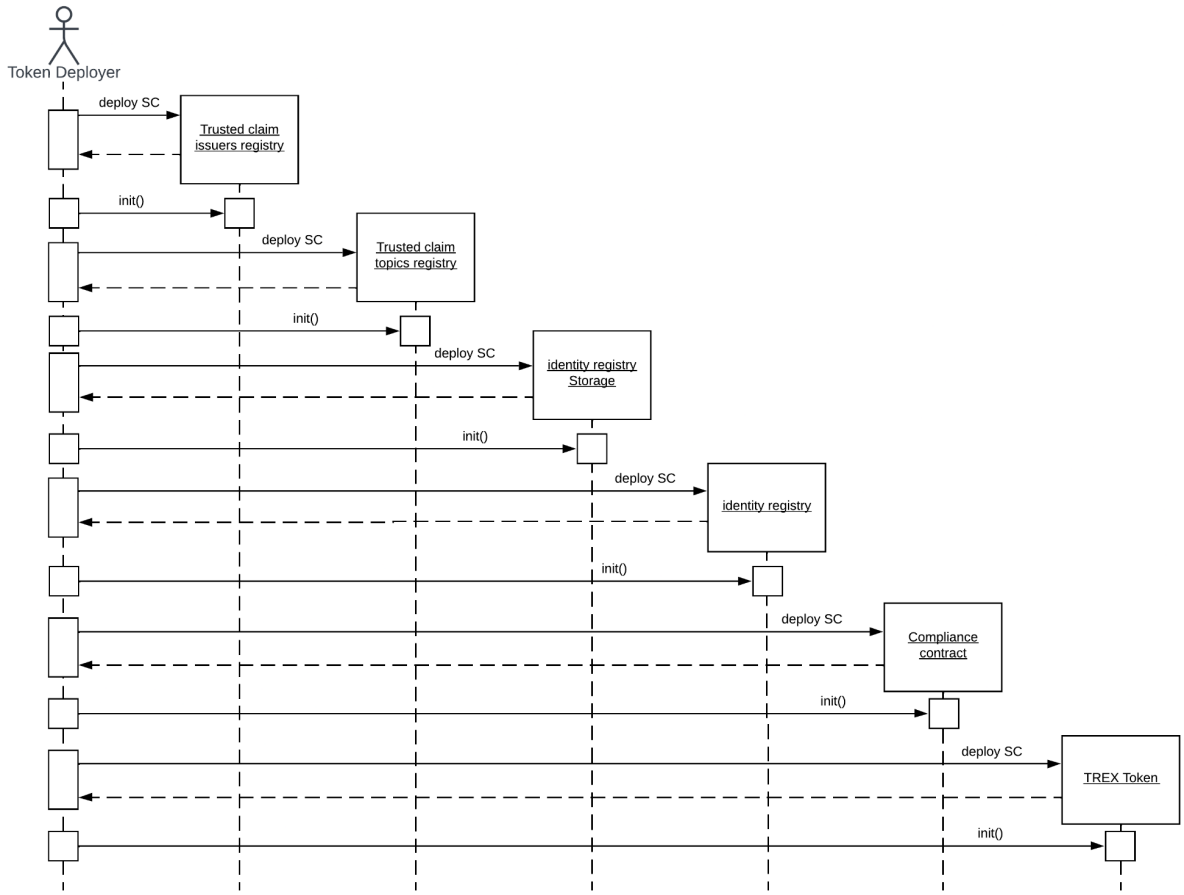

Figure 8 : sequence diagram of standalone security token deployment

1. **Trusted Issuers Registry, Claim Topics Registry and Identity Registry Storage Deployment:** The Token Deployer commences the process by deploying the Trusted Issuers Registry smart contract. This contract houses the addresses of all trusted claim issuers associated with a specific token. At the same time, the Claim Topics Registry contract is deployed. It links to the Trusted Issuers Registry and maintains a list of all trusted claim topics related to the security token. Additionally, the Identity Registry Storage contract is deployed. This contract will later be used to store the records of the Identity Registry.
2. **Identity Registry and Compliance Smart Contract Deployment:** Next, the Identity Registry contract is deployed, utilizing the address of the Identity Registry Storage from the previous step as part of its constructor arguments. This contract holds the identity contract addresses of all eligible users authorized to hold the token. It is responsible for claim verification and interacts with both the Trusted Issuers Registry and Claim Topics Registry contracts. Simultaneously, the Compliance smart contract is deployed. This contract independently operates to check whether a transfer is in compliance with the established rules for the token.
3. **Identity Registry Storage Binding:** Once the Identity Registry is successfully deployed, it is bound with the previously deployed Identity Registry Storage contract. This binding ensures that all identity records are appropriately stored and managed.
4. **Security Token Deployment:** The final step involves the deployment of the Security Token contract. This contract interacts with the Identity Registry to check the eligibility status of investors, enabling token holding and transactions. The Token's constructor function requires the addresses of both the Identity Registry and the Compliance smart contracts, necessitating their deployment before the token contract.

Upon the completion of these steps, the newly deployed token is ready for interaction with validated investors. This sequence guarantees a secure, organized, and compliant setup for the issuance, management, and transfer of security tokens within the T-REX ecosystem.

T-REX Factory deployment

Role: Factory Deployer

The deployment of the T-REX Factory is a pivotal process within the T-REX protocol, demanding precise execution of each step to establish an efficient system for issuing and managing security tokens on the blockchain. The Factory Deployer, the role leading this process, carries out an array of operations, ranging from deploying all necessary smart contracts that constitute the T-REX suite to setting up the Implementation Authority, ensuring robust management of the versions of smart contracts used by proxies.

In this process, we delve into the steps undertaken by the Factory Deployer in deploying the T-REX Factory along with its associated library contracts.

These steps are integral for the proper functioning of the T-REX suite:

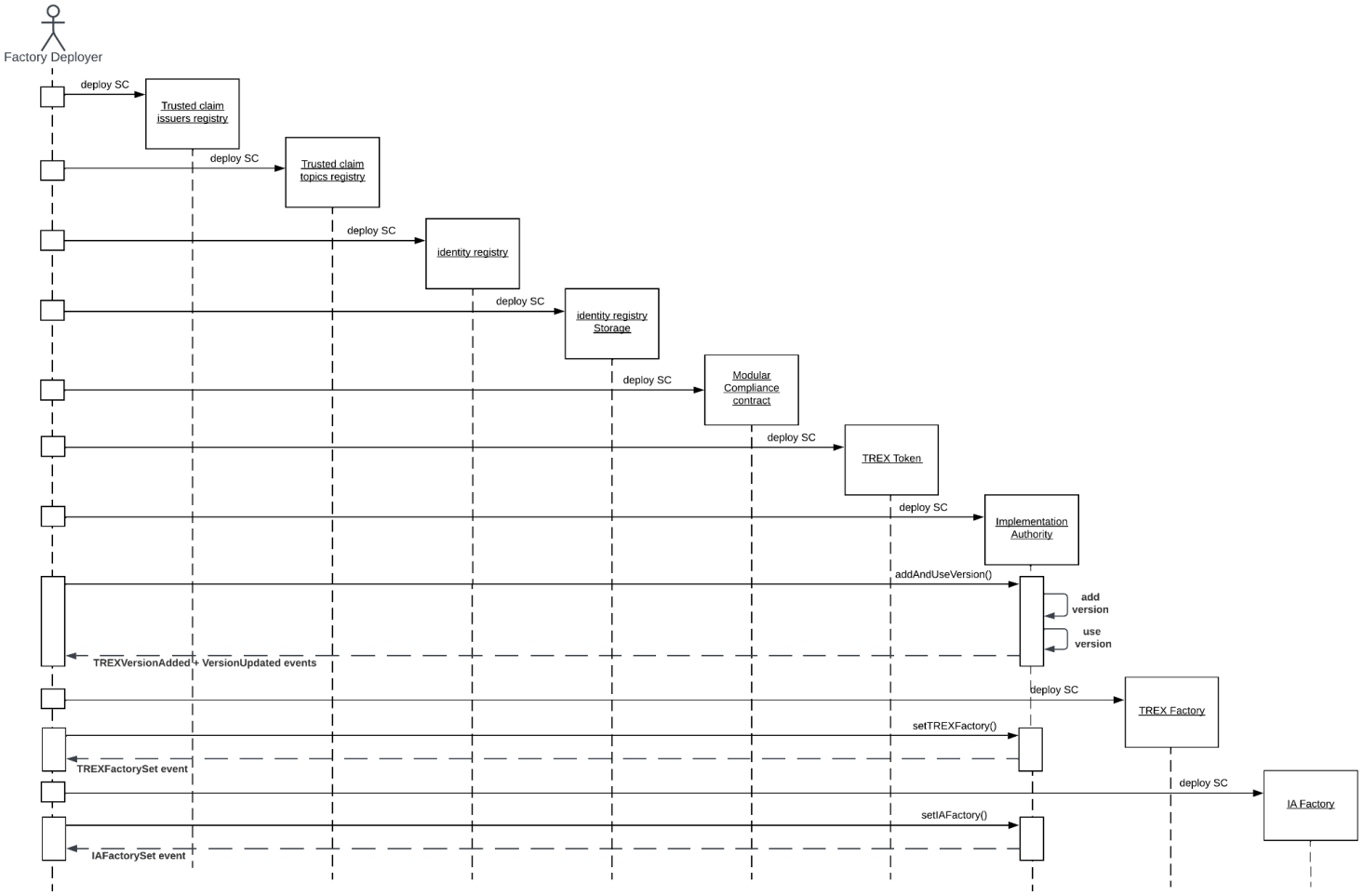

Figure 9 : sequence diagram of T-REX Factory deployment

1. **Library Contracts Deployment:** Kick-starting the process, the Factory Deployer deploys all the integral smart contracts that constitute the T-REX suite. This includes the Trusted Identity Registry, Claim Topics Registry, Identity Registry, Identity Registry Storage, Modular Compliance, and Token contracts.
2. **Implementation Authority Deployment:** Up next, the Factory Deployer establishes the Implementation Authority contract, a crucial component that manages the versions of

smart contracts utilized by the proxies. Upon deployment, the Factory Deployer becomes the contract Owner and can exclusively call functions scoped for the owner. The contract constructor requires the addresses of factories (which are yet to be deployed) and a boolean value indicating whether the contract is the reference contract (the one used by the T-REX Factory). As it is the Implementation Authority for the T-REX Factory, the boolean value should be set as true. The not-yet-deployed factories can be set as a null address for now, to be updated later in the process.

3. **Setting Up the Implementation Authority:** In this step, the Factory Deployer invokes the function addAndUseVersion() to include the earlier deployed library contracts as the current version. It's important that the version aligns with the one in the Token Storage, which can be fetched by calling the version() function on the Token contract.
4. **T-REX Factory Deployment:** Now, the Factory Deployer sets up the T-REX Factory. During the deployment, the address of the Implementation Authority, deployed in step 2, must be incorporated into the contract constructor.
5. **Linking T-REX Factory to Implementation Authority:** Next, the Factory Deployer calls the setTREXFactory() function on the Implementation Authority contract, integrating the T-REX Factory address established in the previous step.
6. **IA (Implementation Authority) Factory Deployment:** The Factory Deployer now deploys the IA Factory contract, enabling Token Issuers who wish to maintain control over their token upgrades to do so autonomously rather than adhering to the general upgrade plan set by the T-REX Factory manager. This IA Factory gets invoked when the changeImplementationAuthority() function is called, which deploys a new non-reference IA contract for their token, or links an existing non-reference IA to their tokens.
7. **Setting IA Factory on the Implementation Authority:** This crucial step enables the reference Implementation Authority to generate new IA contracts. Here, the Factory Deployer calls the function setIAFactory() on the Implementation Authority to record the address of the just-deployed IA Factory.
8. **Final Step - T-REX Deployments:** The Factory is now primed for T-REX deployments, but this can only be performed from the Owner address. To improve security and efficiency, it's recommended not to use a wallet for managing the ownership but instead an external contract that manages roles for granting T-REX deployment rights.

Remember, the deployment process is as significant as the final deployment itself. Each step is designed to maximize the efficiency and functionality of the T-REX suite, making it a dependable platform for issuing and managing security tokens on the blockchain.

Security Token Deployment - Proxy approach using the Factory

Role: Token Deployer

The T-REX protocol facilitates the compliant issuance and transfer of security tokens on the blockchain. A crucial aspect of this is the creation of a T-REX suite, a collective term for the set of smart contracts that govern a security token. The deployment of this suite, executed entirely in a single blockchain transaction, is orchestrated by the T-REX Factory. In the following process description, we outline the steps involved in this intricate deployment procedure, which includes the creation of essential smart contracts such as the Trusted Issuers Registry (TIR), Claim Topics Registry (CTR), and Modular Compliance (MC), as well as the assignment of critical roles to identified agents. Furthermore, this process employs upgradeable proxies, ensuring contract longevity and cross-chain consistency. This deployment process ultimately culminates in a fully functional, compliant, and owner-controlled T-REX suite, ready for the issuance and management of security tokens.

1. **Pre-Deployment Verification:** This process commences by validating the inputs for the creation of a new T-REX suite. The factory deployer ensures that the provided token hasn't been deployed already, and that the claim and compliance patterns adhere to the requisite structure.
2. **Proxy Contracts Deployment:** Following validation, the core suite of smart contracts, i.e., Trusted Issuers Registry (TIR), Claim Topics Registry (CTR), and Modular Compliance (MC) are deployed as upgradeable proxies. Additionally, depending on whether an Identity Registry Storage (IRS) is already specified, a new one is deployed or the existing one is reused. Furthermore, an Identity Registry (IR) is deployed, associating the relevant contracts (TIR, CTR, IRS) with it, and then the Token contract is deployed. The proxy pattern enables upgradeability and consistency of contract addresses across all EVM-compatible blockchains, courtesy of the ERC-1822 standard and the use of create2 opcode.
3. **Claim Topics and Trusted Issuers Setup:** Once the core contracts are deployed, the factory deployer populates the Claim Topics Registry with the necessary claim topics and sets up the Trusted Issuers Registry, associating each trusted issuer with their relevant claims.
4. **Identity Registry Configuration:** The Identity Registry is then bound to the IRS, and the relevant parties (including the newly deployed Token contract) are granted agent roles within the Identity Registry, allowing them to execute particular contract functions.
5. **Modular Compliance Configuration:** The Modular Compliance contract is configured to incorporate all necessary compliance modules, along with their respective settings. This step ensures the compliance rules are set up according to the specifics of the new T-REX token.
6. **Token Deployment Record:** The factory maintains a record of all deployed tokens to prevent duplication and facilitate tracking.

7. **Ownership Assignment:** Upon successful deployment and configuration, ownership of all the deployed contract proxies is transferred to the designated owner, ensuring control over all contract functionalities is in the right hands.
8. **T-REX Suite Deployment Notification:** The process concludes with the emission of a TREXSuiteDeployed event, signaling the successful deployment of the T-REX suite and sharing the addresses of the deployed contracts along with the unique salt used in the process.

Updating trusted claim issuers and trusted claim topics

Role: ABC Corporation

ABC Corporation is responsible for curating the Trusted Claim Issuers and Trusted Claim Topics within the T-REX ecosystem, according to their specific requirements. They can add, update, or remove claim issuers and topics to ensure the eligibility of token holders.

- **Trusted Claim Issuers Registry:** ABC Corporation owns the Trusted Claim Issuers Registry, where they manage the claim issuers and their activities. Suppose ABC Corporation trusts Luxtrust as a claim issuer for specific claim topics. In that case, they can add Luxtrust's identity contract to the Trusted Claim Issuers Registry, along with the array of claim topics for which Luxtrust is trusted. ABC Corporation maintains the flexibility to delete, update, and add Trusted Claim Issuers in the registry as their needs evolve.
- **Trusted Claim Topics Registry:** Additionally, ABC Corporation owns the Trusted Claim Topics Registry, where they manage the claim topics that determine an investor's eligibility as a token holder. ABC Corporation can add or remove multiple claim topics from this registry. For example, if they require investors to complete a KYC process, they can add the corresponding claim topic index (e.g., KYC = 42) to the registry. Other claim topics may include verified address proofs, valid accreditation certificates, etc.

Upon populating the Trusted Claim Issuers and Trusted Claim Topics in the T-REX ecosystem for their token, an investor's identity contract must contain claims listed in the claim topics registry and issued by the trusted claim issuers to qualify as an eligible token holder. This process ensures a compliant environment where all investors meet the necessary requirements and abide by the established rules governing the security token.

Creation of identities on the blockchain

There are two roles that are needed to deploy identity contracts on the blockchain in the T-REX ecosystem:

- Claim Issuers.
- Investors/Token Holders.

An identity can be created by different processes:

- An entity can self-deploy an ERC-734/735 compliant Smart Contract (such as ONCHAINID) directly on any EVM blockchain or by using a decentralized identity management solution;
- An ERC-734/735 contract can be deployed on behalf of an entity by a third party (for example an exchange platform or a token issuance platform creating identities for their users in order to have them subsequently populated with claims related to token eligibility).

Role: Claim issuers

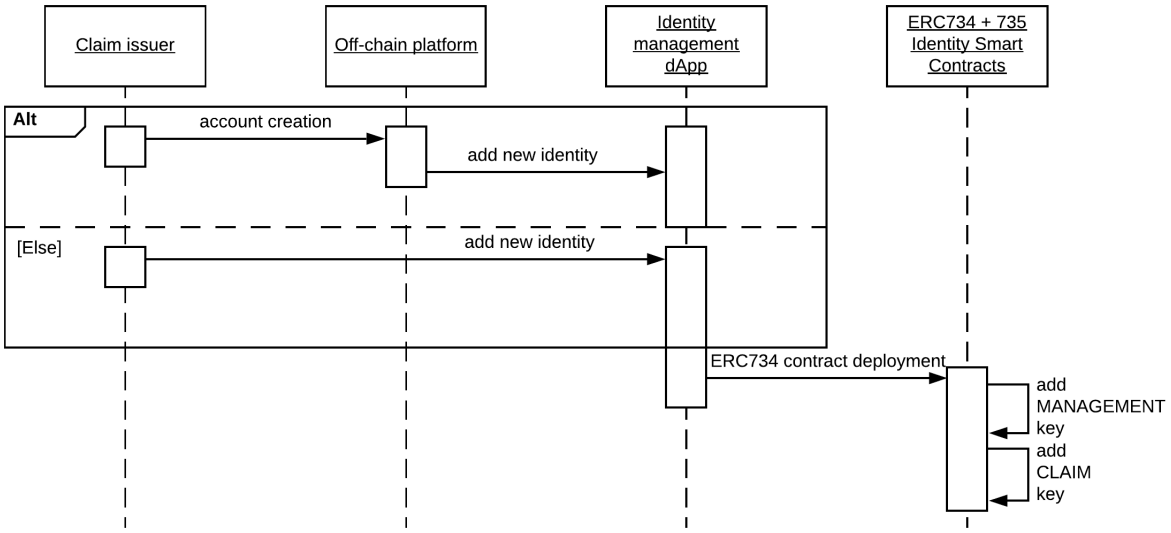

Figure 10 : claim issuer's identity creation

1. Claim issuer deploys an identity contract by any of the methods mentioned above.
2. It then adds a claim signer key to the identity contract deployed. This claim signer key can now be used to sign claims requested by other identities. If this key is deleted, the claim issuer won't be able to issue new claims on this identity.
3. The claim issuer then provides the identity contract address to ABC corporation who in turn adds this address to their trusted issuers registry. (If ABC corporation wants its investors to have a claim issued by this claim issuer).

Role: Investors

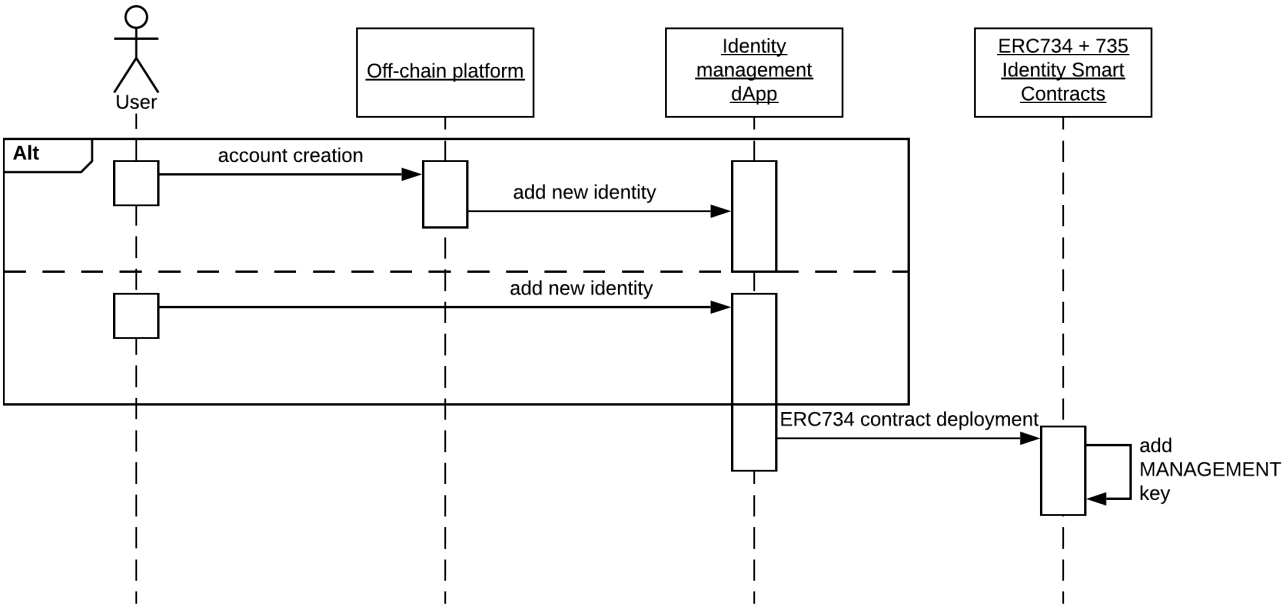

Figure 11 : User's identity creation

A user can deploy his/her own identity contract using our T-REX dapp (or other identity providing services) or can be deployed on behalf of the user by 3rd party entities such as exchanges. On deployment of the ONCHAINID, a management key is added by default which is crucial in the management of keys and claims in the contract.

Claim addition

In the T-REX ecosystem, claims play an essential role by validating the identities and credentials of investors. They are signed attestations issued by a trusted claim issuer that confirm certain attributes or qualifications of the token holders, such as their identity, location, investor status, or KYC/AML clearance. These claims are stored within the investor's ONCHAINID, ensuring compliance with legal and regulatory requirements for the trading of security tokens. The process of adding these claims can be achieved through various approaches, each offering different levels of directness and on-chain or off-chain operations. In any case, the Claim Issuer's wallet, used to sign the claim, has to be registered in the Claim Issuer smart contract as a KEY (following the ERC-734 principles) of type 1 (management) or 3 (claim signer) to be considered a valid claim.

Role: Investors + Claim Issuers

The claim addition can be done :

- **Direct onchain approach:** In this method, the Claim Issuer directly adds the claim to the investor's ONCHAINID. For this to occur, the investor must grant permission to the Claim Issuer to add claims to their ONCHAINID by incorporating the Claim Issuer's claim key onto their own ONCHAINID.
- **Indirect onchain approach:** This method involves the Claim Issuer requesting an execution on the investor's ONCHAINID (see ERC-734 for details about the execute/approve process). The execution request can subsequently be approved by the ONCHAINID owner, enabling the addition of the claim to their ONCHAINID.
- **Indirect hybrid approach:** Here, the Claim Issuer prepares the claim off-chain (including the Claim Issuer’s signature) and transmits it to the ONCHAINID owner. The owner can then incorporate it into their ONCHAINID.

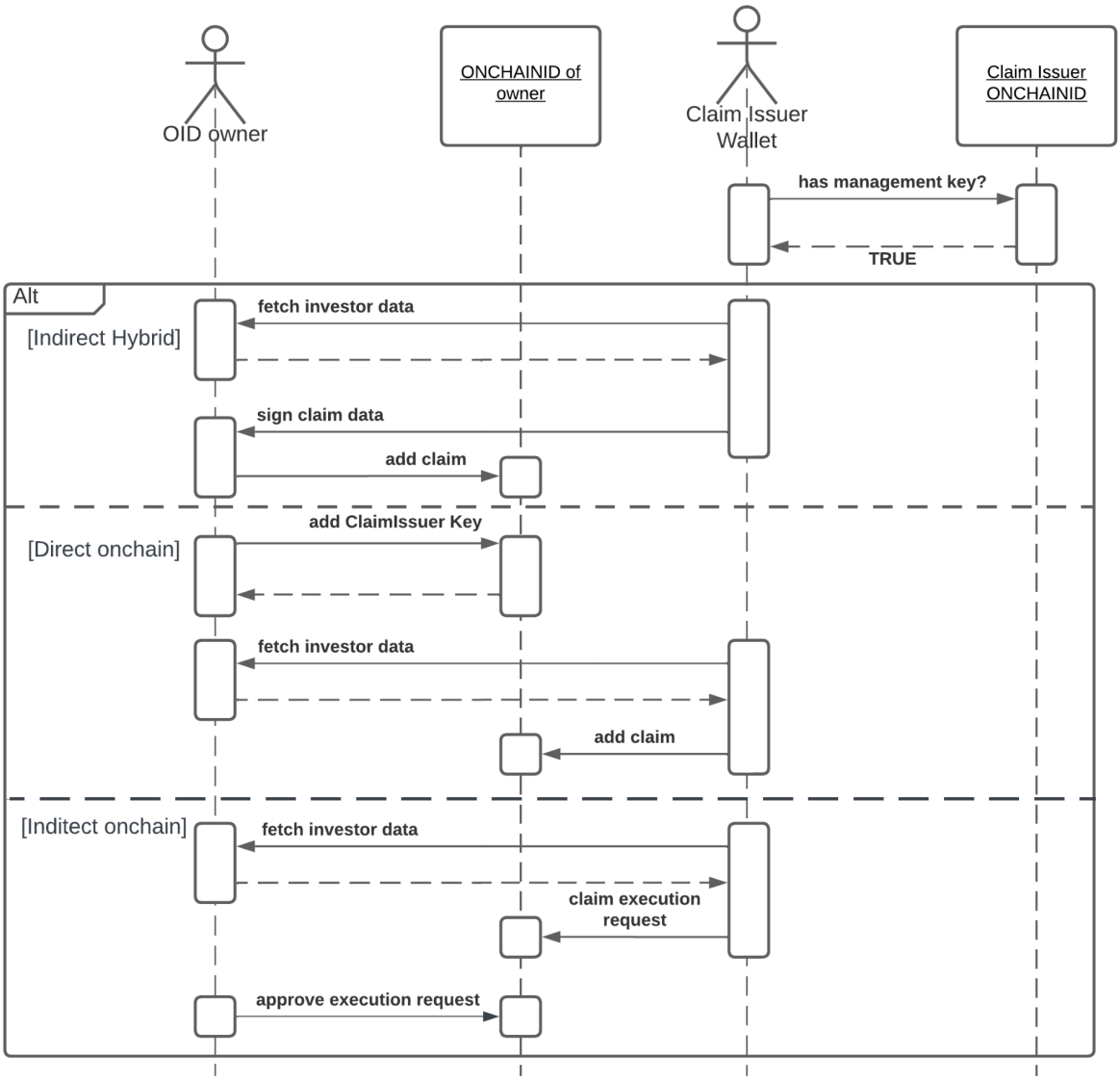

Figure 12 : sequence diagram of a claim addition

User registration in the identity registry

Role: ABC Corporation

Once the investor has an identity contract deployed, and has passed KYC and eligibility checks requested for the token considered, it must be added in the identity registry in order to enable the user to hold/transfer the token. The purpose of this registry is to store the ONCHAINID contract address corresponding to the wallet address that should be able to hold tokens. This contract has the all-important

function - isVerified() which is responsible for the verification of claims in the identity contract corresponding to a user's address. As discussed earlier, the identity registry interacts with the trusted issuers registry to verify claims based on the initial security token requirements.

For security purposes, only the token issuer or his agent can add an identity contract to the registry and only if the owner of the identity contract initiates the request to be added in the identity registry. Also, one wallet address can only have one identity contract stored in the registry.

The token issuer or ABC corporation has full ownership of this contract and is responsible for the management of user identities. To register a new identity, the Issuer has to call the registerIdentity() function in the identity registry contract. The user's wallet address and the ONCHAINID contract address are used as parameters.

Compliant transfer of ownership

The T-REX standard makes sure that the transfer of security tokens is compliant.

Here are the steps T-REX to ensure compliance:

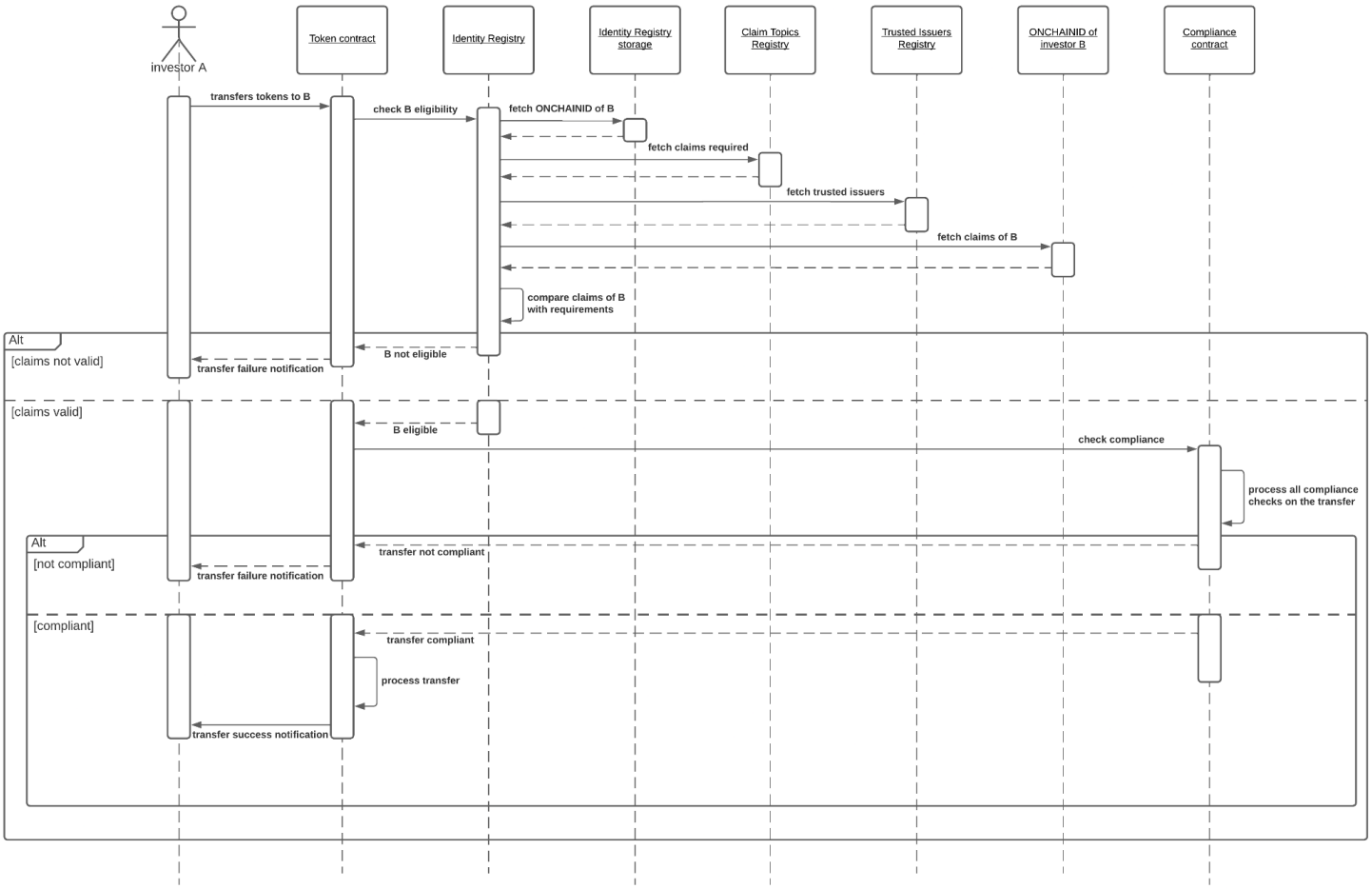

Figure 13 : compliant transfer of ownership of T-REX token

1. User calls the transfer/transferFrom function.
2. The isVerified() function is called from within the transfer/transferFrom functions to instruct the identity registry to check if the receiver is a valid investor (i.e. if his address is in the identity registry of the token - meaning he holds the necessary claims needed to be eligible for this token). If no identity contract is found corresponding to the address of the receiver, an error status is returned.

3. The identity registry fetches the claims in the receiver identity contract and decodes the claims to extract the claim topics and the signatures. It then compares the found claim topics with the claim topics in the trusted claim topics registry. If no matches are found, a failed status is returned and the transfer is not allowed to proceed.
4. If a match is found, then the signature is checked. First, the trusted claim issuers are fetched from the trusted claim issuers registry. Then for each trusted issuer, it is verified whether the signature was done by a claim signer key present in the trusted claim issuer's identity contract. If no matches are found, then a failed status is returned and the transfer is not allowed to proceed.
5. If a match is found, then a success status is returned to the security token, which means that the receiver is eligible.
6. The security token contract then calls the canTransfer function on the Compliance smart contract to verify if the transaction is not violating a compliance rule enforced by the latter.
7. The Compliance smart contract performs all checks on the transaction and returns a status, if that status is negative it means the transfer is not compliant and the whole transaction is rejected by the smart contracts.
8. If the Compliance returns a positive status the transfer of tokens can happen and the token balances are updated on the ledger.

Token ownership on the blockchain

As outlined in the Ethereum Whitepaper[20], Ethereum and other EVM-compatible blockchains are composed of two primary account types: Externally Owned Accounts (EOAs), controlled by private keys, and Contract Accounts, governed by their associated contract code. EOAs are typically managed via wallets such as Metamask or MyEtherWallet. These wallet applications have evolved to accommodate ERC-20 tokens, allowing for the display of token ownership by EOAs and enabling transfers of tokens from an EOA to a designated address.

For seamless integration with these wallets, the model proposed here presumes that the tokens on the blockchain are owned by an EOA that is associated with an ERC-734 contract account. This arrangement introduces a slight increase in complexity compared to a model where the tokens are directly owned by an ERC-734 contract account, which was the original intent behind the creation of ERC-734 and ERC-735 standards. However, as the adoption of blockchain technology grows, it is anticipated that standard wallets will evolve to support ERC-734 Identity tied to an EOA, much like how they have adapted to accommodate ERC-20 tokens and Ethereum Name Service (ENS).

From its inception, the T-REX protocol has supported both models. If the receiver's address of the token transfer is a contract address compliant with ERC-734, the necessary KYC checks are performed as outlined in this document. Conversely, if the receiver's address is an EOA, a search is conducted in the

[20] https://github.com/ethereum/wiki/wiki/White-Paper

Identity Registry to identify if an Identity is linked to this EOA. Following the establishment of this link, the KYC checks are then performed.

Conclusion

The advent of blockchain technology has opened up new possibilities for the financial markets, particularly in the realm of asset tokenization. The T-REX protocol, as outlined in this whitepaper, represents a significant step forward in this direction. By enabling compliant issuance and management of permissioned tokens, T-REX brings unprecedented efficiency, accessibility, and liquidity to the market.

However, the journey towards widespread adoption of tokenized securities is not without challenges. Regulatory compliance, identity management, and ensuring the rights and security of all stakeholders are critical issues that need to be addressed. The T-REX protocol, with its open-source ERC3643 token standard and decentralized validation system, offers robust solutions to these challenges.

As we move forward, it is essential to continue refining and expanding these solutions, keeping pace with the evolving regulatory landscape and the needs of the market. The future of financial markets lies in harnessing the power of blockchain technology, and the T-REX protocol is poised to play a pivotal role in this transformation.