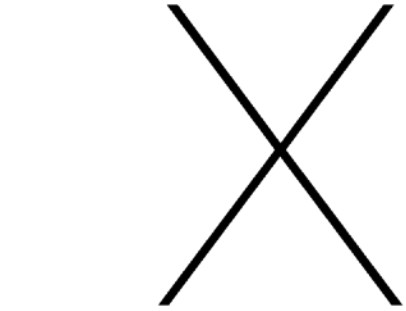


XEN CRYPTO

XENFT

1. XEN Torrent Protocol

Fair Crypto Foundation

Litepaper v0.3

@MrJackLevin @lbelyaev

XENFT - XEN Torrent (XENT)

XEN Crypto

XEN Crypto (XEN) is an EVM compatible ERC-20 token. Devised by Jack Levin and launched by the Fair Crypto Foundation in 2022, XEN Crypto laid the groundwork to create an ecosystem of crypto instruments centered around the First Principles and designed for mass world adoption.

XENFT

As mentioned above, XEN Crypto is implemented as a Fungible Token (FT) adhering to the ERC-20 standard. Its primary goal is to function as a medium of exchange. People can mint, buy, stake and sell XEN using the fast growing partner's ecosystem.

With the launch of XENFT, Fair Crypto Foundation takes a new step towards XEN use and mass adoption. XENFT, is a derivation of XEN NFT, is a non-fungible token that interconnects with XEN ERC20 token by virtue of minting as well as the implementation and use of the Proof of Burn protocol (native to XEN ERC-20 Smart Contract).

This paper is the first part of a series of litepapers dedicated to XENFT and covers XEN Torrent project. Thus any reference to XENFT in the text below refers specifically to XENFT represented by XEN Torrent Smart Contract.

XEN Torrent

XEN Torrent is a multi-faceted instrument which is run by an ERC-721 compatible contract. It can be deployed on any EVM compatible network where XEN Crypto contract has already been operational.

Same as XEN, XEN Torrent adheres to the first principles of crypto:

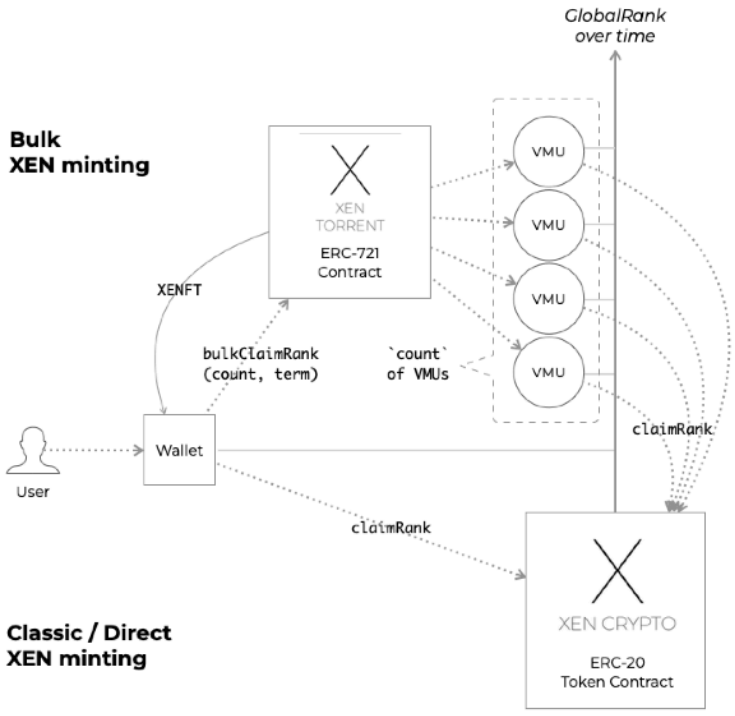
- No pre-mint;
- No whitelists, blacklists or any special allocations;
- Immutable contract;
- No admin (control) keys;

Utility

Under the hood, XEN Torrent automates the execution of a series of on-chain transactions, maximizing XEN (ERC-20) mint by virtualizing Ethereum addresses used to claim cRanks (aka Virtual Minting Units, or VMUs), controlled by a user via XEN Torrent smart contract.

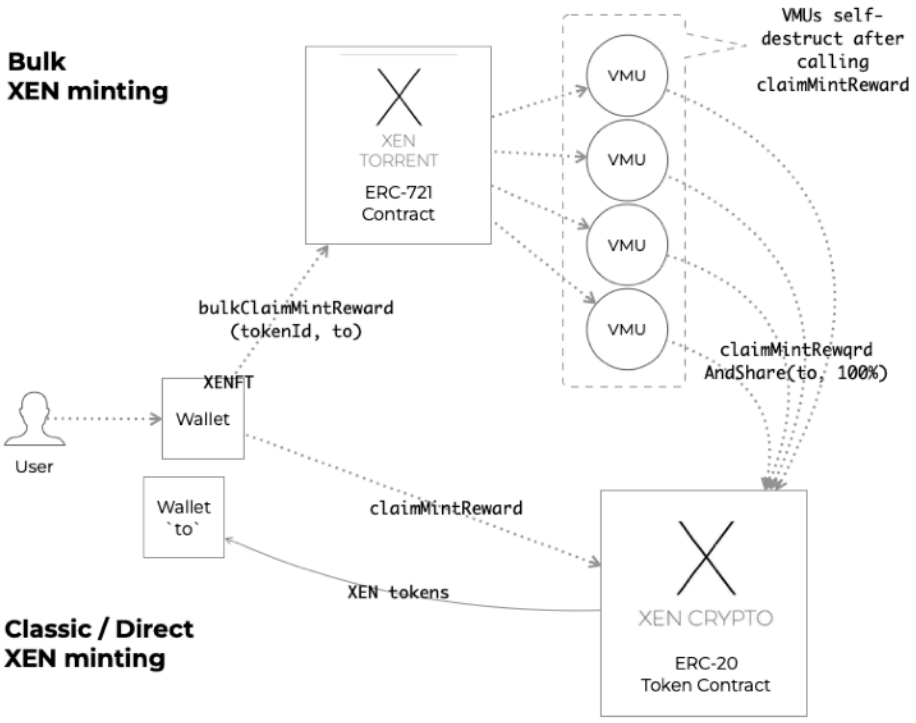
Similar to XEN Crypto, which has a two phase operation (Claim Rank and Claim Mint Reward), XEN Torrent also has two phases.

During the initial phase, a set of VMUs is created, whose number is controlled by the ‘count’ parameter set by a user. Upon creation, each VMU calls the original XEN Crypto smart contract with ClaimRank function, initiating XEN minting. ‘Term’ parameter for ClaimRank is also set by the user. In a nutshell,



XEN Torrent allows to start the parallel minting of XEN from the single user-controlled address, using a single transaction.

Second and final phase is available for a user when XEN minting controlled by XEN Torrent Protocol reaches maturity (the 'term' amount of days passes since the original phase took place). Once the maturity is reached, XEN Torrent Protocol user can perform a bulk ClaimMintReward operation via controlled VMUs (created on the initial phase). XEN tokens minted during this operation are then transferred to a user-designated address (which could be user's original address, or any other address on the network).



Asset

Each user operation with XEN Torrent initiating bulk minting of XEN Crypto issues a non-fungible token (XENFT/ERC-721) which is transferred to the user. Each XENFT is unique and is not mutually-interchangeable with any other XENFT (thus non-fungible).

Any user (represented either via a wallet address or a smart contract) can own an unlimited number of XENFTs. Each newly issued XENFT represents an access control right to the bulk XEN minting operation. In order to claim XEN tokens once the maturity is reached, the user has to be in possession of the XENFT.

Since XEN Torrent tokens adhere to ERC-721 NFT standard, they are transferrable. This means that one user can transfer a XENFT owned by them to another user (whether transfer happens as a result of a sale or swap or free of charge is out of the scope of this paper and XEN Torrent smart contract).

As mentioned above, XEN Torrent XENFT serves as an access token to claim rights to the bulk XEN minting. It's a classic bearer token, which means that whoever owns (presents) XEN Torrent Token can claim the proceeds of XEN ERC-20 mint.

XENFT Properties

Each issued XENFT will have the following properties encoded and stored in the XEN Torrent smart contract:

- XEN Crypto minting related
  - Term (days)
  - Maturity Timestamp
  - cRank (related to the first VMU, or the start of the batch)
  - AMP (related to the first VMU, or the start of the batch)
  - EAA (related to the first VMU, or the start of the batch)
- XEN Torrent specific
  - Count of VMUs
  - Category (see below)
  - Amount of XEN burned (see below)
  - Redeemed (or not)

All the properties captured in the XENFT are immutable, except for the last one. 'Redeemed' prop is a boolean value and is set to 'false' upon XENFT minting. Once the user has performed the second phase and claimed the

minted XEN ERC20 tokens, XENFT is said to be redeemed and the ‘redeemed’ property is set to True.

Important notes on claiming cRank and Minting

- Once XENFT is issued to a user, XEN Torrent smart contract has no control or encumbrance over the token. XEN Torrent smart contract cannot reverse token sale or transfer, freeze the token or perform any operation on it by itself, without explicit instruction from the current XENFT owner.

- XEN minting operation initiated via XEN Torrent smart contract is run by XEN Crypto ERC-20 smart contract; whereby the same rules of claiming rank and claiming mint rewards apply. XEN minting initiated via XEN Torrent has no special treatment, it shares the same Global Rank counter as direct operations via XEN Crypto. Same is true for mint claim/withdrawal penalties; users are encouraged to be mindful of the maturity date and the 7 day withdrawal time window.

Categories of XEN Torrent XENFTs

XEN Torrent XENFTs come in 3 different flavors and categories:

- Rare
- Limited
- Common or Ordinary

Rare

Rare category is the top tier in XEN Torrent hierarchy. Its issuance is limited to the total amount of 10,000 NFT tokens controlled by the smart contract and is immutable. In order to mint a Rare XENFT, the following criteria must be met:

- Total Supply of issued Rare XEN Torrent XENFTs is less or equal to 10,000,
- Count of VMUs in the bulk XEN mint is 100 or more,

- User is in possession and is willing to burn XEN tokens for the privilege to mint one of the Rare XENFTs

Amount of XEN tokens to be burned for the Rare XENFT determines the rarity class (inside the Rare category), each of which has its own limits of issued tokens:

- Class 1: 100 of Rare XENFTs (token IDs 1...100)
- Class 2: 900 of Rare XENFTs (token IDs 101...1,000)
- Class 3: 2,000 of Rare XENFTs (token IDs 1,001...3,000)
- Class 4: 3,000 of Rare XENFTs (token IDs 3,001...6,000)
- Class 5: 4,000 of Rare XENFTs (token IDs 6,001...10,000)

Specific amounts of XEN which is required to mint one of the XENFTs of each class, is not defined until the XEN Torrent contract deployment date and could be different for each network where XEN Torrent is deployed. However, the rule of thumb is that each lower class will require a declining scale of XEN burn compared to the higher one.

Limited

Limited category, unlike the Rare category, is not limited by the number of tokens issued but rather is limited by time. Limited XENFTs will be issued to any number of qualified users according to following the criteria:

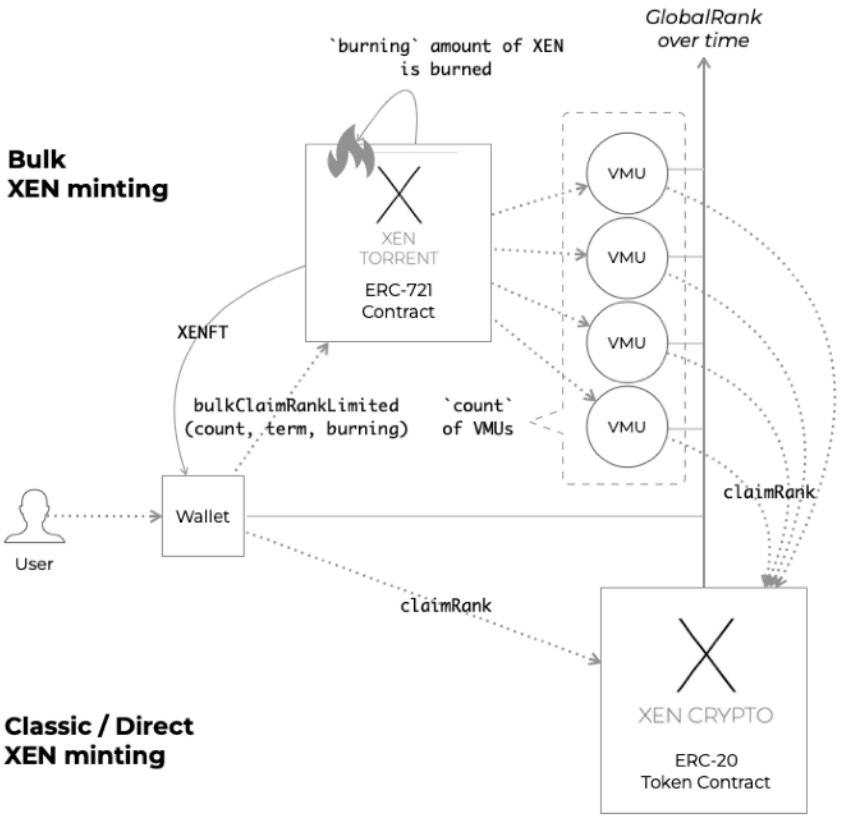
- Time of Limited XENFT issuance is 365 days (31,536,000 seconds) since the moment of XEN Torrent smart contract deployment (which is captured by the `genesisTs` immutable variable),
- Count of VMUs in the bulk XEN mint is 100 or more,
- User is in possession and is willing to burn XEN tokens for the privilege to mint one of the Limited XENFTs.

Just like with the Rare XENFT category, the specific amount of XEN burn which is required to mint one of the Limited XENFTs is to be defined until the XEN Torrent contract deployment date and could be different for each network where XEN Torrent is deployed. However, this amount will be less than the amount required by the lowest class of Rare XENFT category.

There is no limit to how many Limited XENFTs each of users can hold.

Token IDs for Limited XENFTs share the same counter with Ordinary / Common XENFTs (token IDs start at 10,001).

The difference between minting Rare or Limited categories of XENFTs with regards to burning XEN is shown in the diagram below.



Ordinary / Common

XENFTs of this category can be minted free by anyone, at any time, any number of times. The only commitment from a user (universal for all XENFT categories) is to pay the network gas fee for the initial XENFT mint transaction. XEN (ERC-20) burn isn't required.

Common XENFTs can be minted to represent as low as 1 VMU with the term as low as 1 day.

In order to make Common XENFTs ‘less common’ and introduce additional game theory spirit for all XENFTs owners, XEN Torrent Common tokens are also divided into classes. In this case the qualifying parameter is XENFT’s `power` which is computed as:

$$Power = |VMU| * term,$$

Or, in plain terms, it’s the count of launched VMUs times XEN minting term (expressed in days).

The choice of this synthetic property is determined by the fact that it serves as the best proxy for the value of future XEN rewards, and could be used as a quick (albeit very approximate) comparison factor between two different XENFTs.

Classes of Common XENFTs are defined as:

$$C_i = \lfloor Power/7500 \rfloor$$

Accordingly (by class index computed as shown above):

- Class 0 (Ruby): Power 1...7,500
- Class 1 (Opal): Power 7,501...15,000
- Class 2 (Topaz): Power 15,001...22,500
- Class 3 (Emerald): Power 22,501...30,000
- Class 4 (Aquamarine): Power 30,001...37,500
- Class 5 (Sapphire): Power 37,501...45,000
- Class 6 (Amethyst): Power 45,001...52,500
- Class 7 (Xenturion): Power 52,501...

Note that Class 7 of Common XENFTs is unbounded with regards to the `power` parameter; any XENFT with power above 52,501 will be classified as Xenturion.

XENFT Metadata Art

It is Fair Crypto Foundation's belief that XENFT value will be determined by both its utility and its rarity (category and class) properties.

Metadata art is a very important manifestation of the XENFT value, therefore all token categories and classes will bear distinct art capturing various aspects of the XENFT value.

(All examples below are for illustrative purposes; designs could change before actual deployment day).

Example of a Rare XENFT cover. Note the distinct translucent colors and Ying-Yang mark denoting Rare category.

Example of a Limited XENFT cover art. Note the distinct translucent colors and burn mark denoting Limited category.

Example of various classes of Common XENFT cover art are shown below. Note the different colors / class names and Pick Axe mark denoting Common category.

General info on XENFT metadata properties

XEN Torrent contract supports ERC-721 extension which allows anyone to query XENFT metadata for any token ID via `tokenURI` method. The said method returns a JSON-formatted string that contains various properties of XENFTs, including URL to retrieve unique XENFT image.

Following the principles of decentralization, XEN Token contract is generating all metadata including unique metadata art image on-chain. Image is returned as data URL base64-encoded string generated by smart contract.

XENFT and XEN Crypto

As mentioned above, XEN Torrent and XEN Crypto work together in concert:

- XEN Crypto contract address is captured in the XEN Torrent smart contract at the moment of its deployment and is immutable;

- When required by XEN Torrent contract for minting XENFTs of Rare and Limited categories, XEN Token contract burns of all of the approved XEN tokens via `Proof-Of-Burn` mechanism which is a part of XEN Crypto protocol;

- XEN Torrent contract (via VMUs) uses XEN Crypto protocol to claim rank and then mint XEN tokens;

N.B. Neither of XEN Crypto or XEN Torrent contracts ever own any of their assets. All XEN tokens and XENFTs are always owned by users.

Secondary sales of XENFT

As mentioned above, XEN Torrent implements ERC-721 token standard, which allows for tokens (XENFTs) to be transferred between users without any limitations.

The nature of deals between users that result in XENFT ownership being transferred between accounts is well outside the scope of this light paper, however, there are several important things to keep in mind:

- Any transfer of XENFT ownership is either seller-initiated or buyer-initiated,

- A seller-initiated transfer (transfer between owned accounts, 'gifting' of XENFTs, etc. could be accomplished via single transaction - a call to `transferFrom()` function, where current owner states token ID and the token transferee (`to` address).
- A buyer-initiated transfer (commonly occurring in as 'sale' transaction, typically via some sort of NFT marketplace like Opensea, Rarible, etc.) is typically a 2-step process:
  - 1st step is `approve` function call, giving `power-of-attorney` right to a 3rd party agent (typically a marketplace) to make future transfer
  - 2nd step is the `transferFrom` transaction call which lists addresses of `from` and `to` subjects and the token ID.

N.B.:

- `transferFrom` transaction without prior `approve` transaction will fail if initiated by a third party!
- Users need to be careful when issuing an `approve` transaction since they are basically giving away the right to a 3rd party to command their NFT(s)
- Same as with power-of-attorney, `approve` powers could be reworked before the transfer of ownership occurred
- If XENFT current and/or new owner is not a wallet but rather a smart contract, such contract should be specially designed to be able to hold ERC-721 tokens. If such contracts don't support it, transfer of an XENFT to them will fail!

Technical Details

XEN Torrent Contract Structure

XEN Torrent is a ERC-721 compatible smart contract that is based on widely accepted and well-tested OpenZeppelin reference implementation.

Due to a large amount of code and given 24 Kb byte code limit for EVM smart contracts, XEN Torrent is broken into separate parts (libraries):

MintInfo

`MintInfo` defines a structure for recording all elements related to a separate bulk mint operation (a Torrent). In the interest of saving gas, `MintInfo` packs 7 different properties into a single uint256 store variable. `MintInfo` library has convenience methods for encoding and decoding of a single record, as well as accessors for separate properties of a record.

SVG

`SVG` is a library that is responsible for producing of a unique image corresponding to each XENFT (examples were shown above). It defines several structured types that are used to pass parameters to SVG (data params, Color encoding params, Gradient params). The only externally accessible method of this library is `image()`, which returns a byte string of an SVG image.

StringData

`StringData` is a small library that deals with storing and accessing of XEN quotes (used in XENFT metadata art) and names of different XENFT series.

DateTime

`DateTime` is a library for converting Unix epoch timestamps (number of seconds elapsed since Jan 1, 1970) to human readable Date and Time strings (in UTC timezone). It is based on an open-sourced library BokkyPooBah's

DateTime Library v1.01. Main function exported by DateTime library is `asString(unit256 ts)` which returns string representation of a supplied timestamp.

MetaData

`MetaData` library contains the main building blocks for producing XENFT metadata, including image building and creation of a JSON-encoded object with all XENFT properties. It exports 2 functions: `svgData(…)` and `attributes(uint256 count, uint256 mintInfo).`

XENTorrent

`XENTorrent` is the main smart contract that extends ERC-721 boilerplate and implements specific logic and stores data related to XEN Torrent protocol.

Constructor of XENTorrent has the following signature:

```
constructor(address xenCrypto_, uint256[] memory
burnRates_, uint256[] memory tokenLimits_)
```

It accepts and stores several important immutable values used in XEN Torrent protocol:

- address of original XEN Crypto contract (`xenCrypto_`)
- XEN burn rate parameters used to mint Rare and Limited XEN Torrent NFTs (`burnRates_`)
- Limits of each of Rare and Limited XEN Torrent series (`tokenLimits_`)

Public interface of XEN Torrent smart contract consists of the following methods:

Read only

`owner()` - returns the deployer address for the XEN Torrent contract. Used to set up NFT collections params on Opensea and other marketplaces.

`ownedTokens()` - returns an array containing tokenIds of all XEN Torrent tokens for the current user (who calls this method).

`tokenURI()` - returns a data URL- referred JSON-encoded string containing required metadata object for each unique XENFT.

Write (transactions)

`bulkClaimRank(…)` starts bulk XEN minting operation with `count` VMUs and for `term` days and issues XENFT to the calling user.

`bulkClaimRankLimited(…)` starts bulk XEN minting operation with `count` VMUs and for `term` days and issues XENFT to the calling user. Is used to request a Rare of a Limited XENFT (via `burning` param that specifies amount of XEN to be burned). As mentioned above, the specified amount of XEN is burned on success of this call.

`bulkClaimMintReward(…)` terminates current bulk XEN minting operation, claims XEN tokens and destructs all VMUs. Access to this method is limited to XENFT `tokenId` owner. Successful call changes the state of XENFT to redeemed = true.

VMU Implementation

VMU (Virtual Minting Unit) is an on-chain wallet address that is being created by the XEN Torrent contract. Each VMU is itself a smart contract that is controlled by a user via XEN Torrent contract (since smart contracts don't have private keys and therefore cannot sign transactions by themselves).

VMU smart contract is created according to a `Minimum Proxy` pattern (EIP-1167) which allows to cheaply clone an existing contract - in our case XEN Torrent Contract. For security, in order to distinguish between the original XEN Crypto and its clones (VMUs), original contract address is recorded in itself via immutable variable set in constructor.

External interface of every VMU consists of the following methods:

`callClaimRank()`

`callClaimMintReward()`

`powerDown()`

First 2 methods make proxy calls to XEN Crypto contract to claim rank and claim mint correspondingly.

The last method makes VMU contract self-destruct.

All these methods are callable only by the original XEN Torrent smart contract, which allows effective access control for a user to start bulk mint and to terminate it at maturity.

Burning XEN and Preventing Reentrancy

As mentioned above, getting Rare or Limited category of XEN Torrent XENFT requires a user to burn a certain amount of XEN tokens. In order to be atomic (burned XEN in exchange for minted XENFT), this transaction requires a two phase communication between XEN Torrent and XEN Crypto contracts.

Phase 1. Spin VMUs and bulk claim rank. On completion, XEN Torrent calls `burn()` method of XEN Crypto contract with details of token owner and token quantity.

Phase 2. Once XEN tokens are burned by XEN Crypto contract, it calls back XEN Torrent contract via `onTokenBurned()` method. This method completes the atomic transaction, mints XENFT and records its MintInfo into the contract storage.

In order to maintain state between the 2 phases, and also to prevent reentrancy attacks, a private variable `_tokenId` is used. It's set in phase 1 and cleared in phase 2. If the contract would receive either a call to initial phase 1 method while `_tokenId` is non-zero, OR if a callback would be received with `_tokenId` is zero, such transaction would fail.

Proof-of-Burn Protocol

Same as XEN Crypto, XEN Torrent supports Proof-of-Burn protocol, which allows any 3rd party smart contract to burn XENFTs - for example in exchange for some other crypto token (or tokens).

Proof-of-Burn protocol consists of 2 parts:

1. Main contract (token controller) implements method `burn()` callable by another contract (token agent). As in the case with burning XEN described above, calling this method initiates Proof-of-Burn transaction.

2. Token agent contract implements `IBurnRedeemable` interface which includes `onTokenBurned()` method and `Redeemed` event. This contract also needs to implement ERC-165 standard, specifically it's `supportsInterface()` method which shall respond affirmatively to requests about supporting `IBurnRedeemable` interface. Token controller calls `onTokenBurned()` method and emits `Redeemed` event once the burn procedure is complete.

N.B.:

- If the Token agent contract doesn't support `IBurnRedeemable` or doesn't advertise it via ERC-165, the Proof-of-Burn transaction will fail

- For burn transaction to succeed, Token agent contract must be approved as an operator for a specific XENFT token ID via ERC-721 standard `approve` method, otherwise transaction will fail.