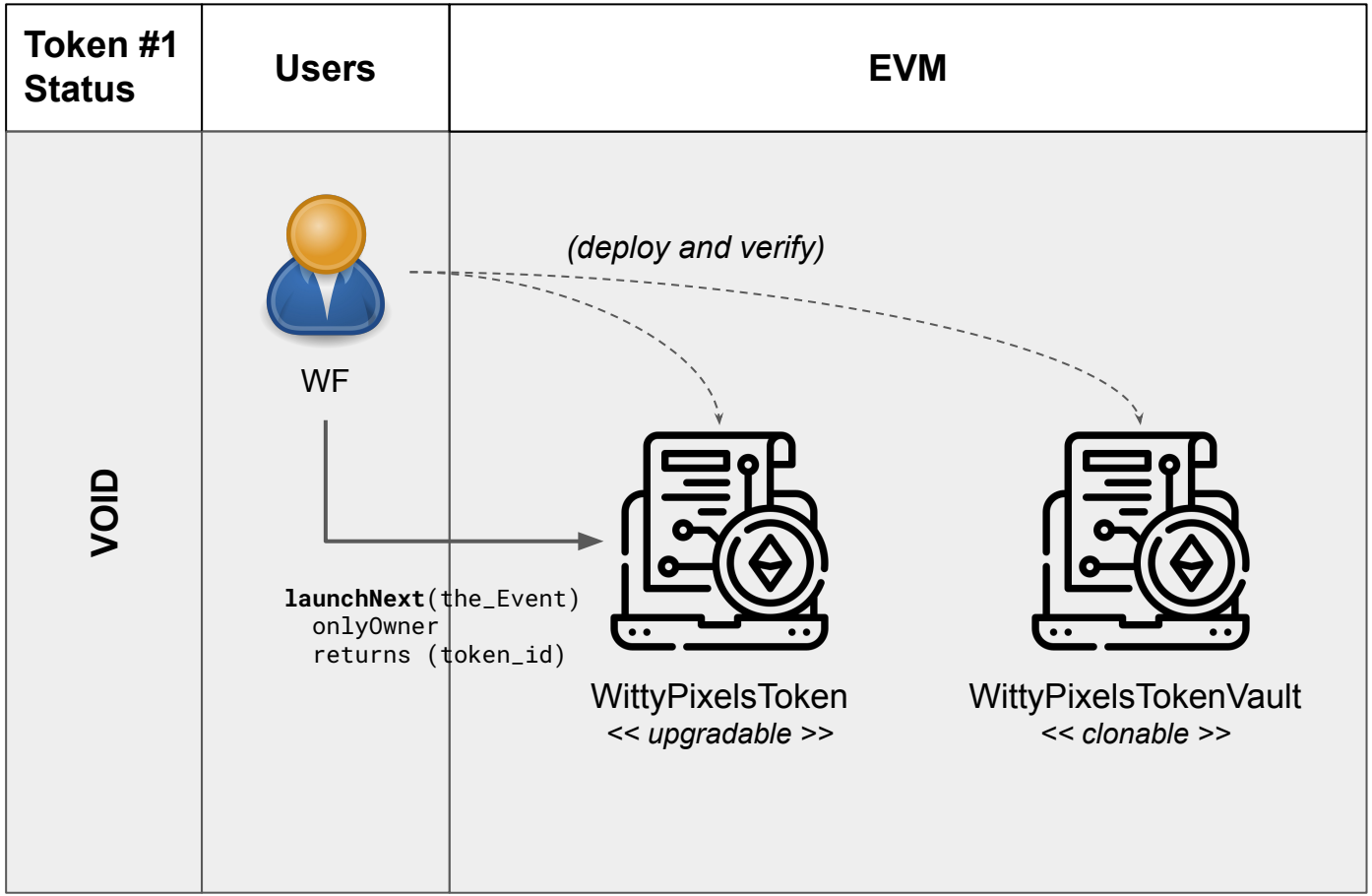
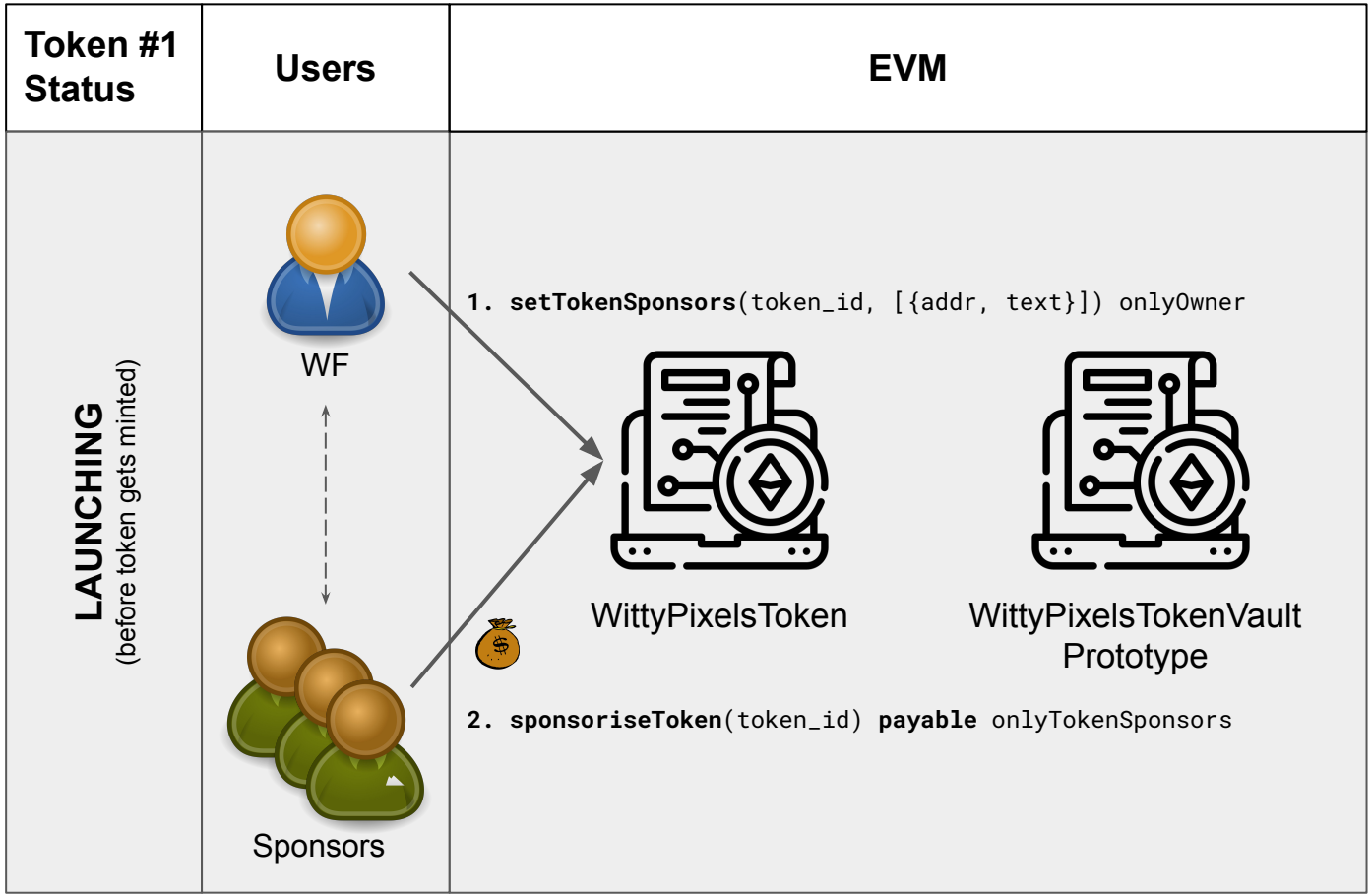
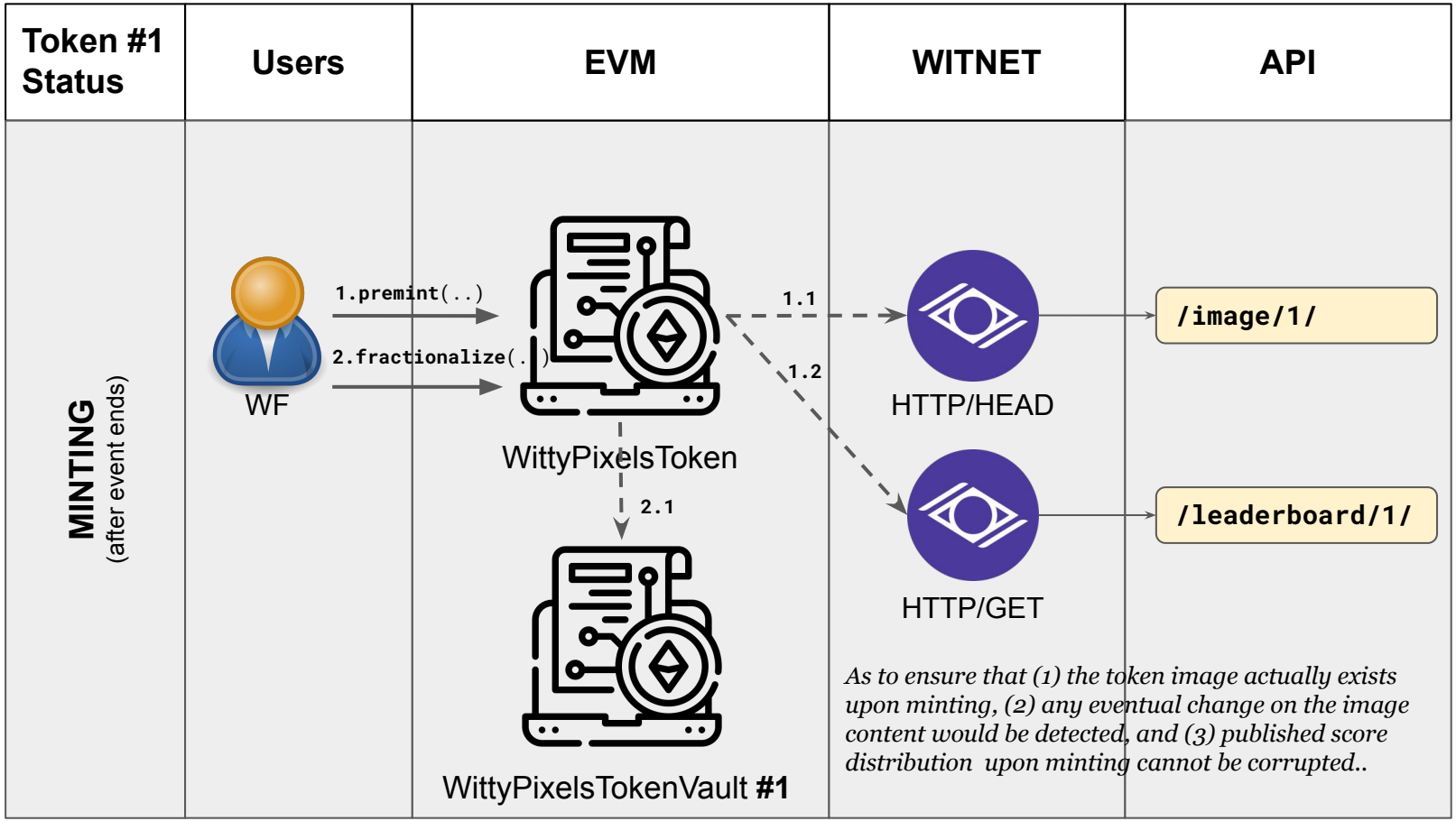
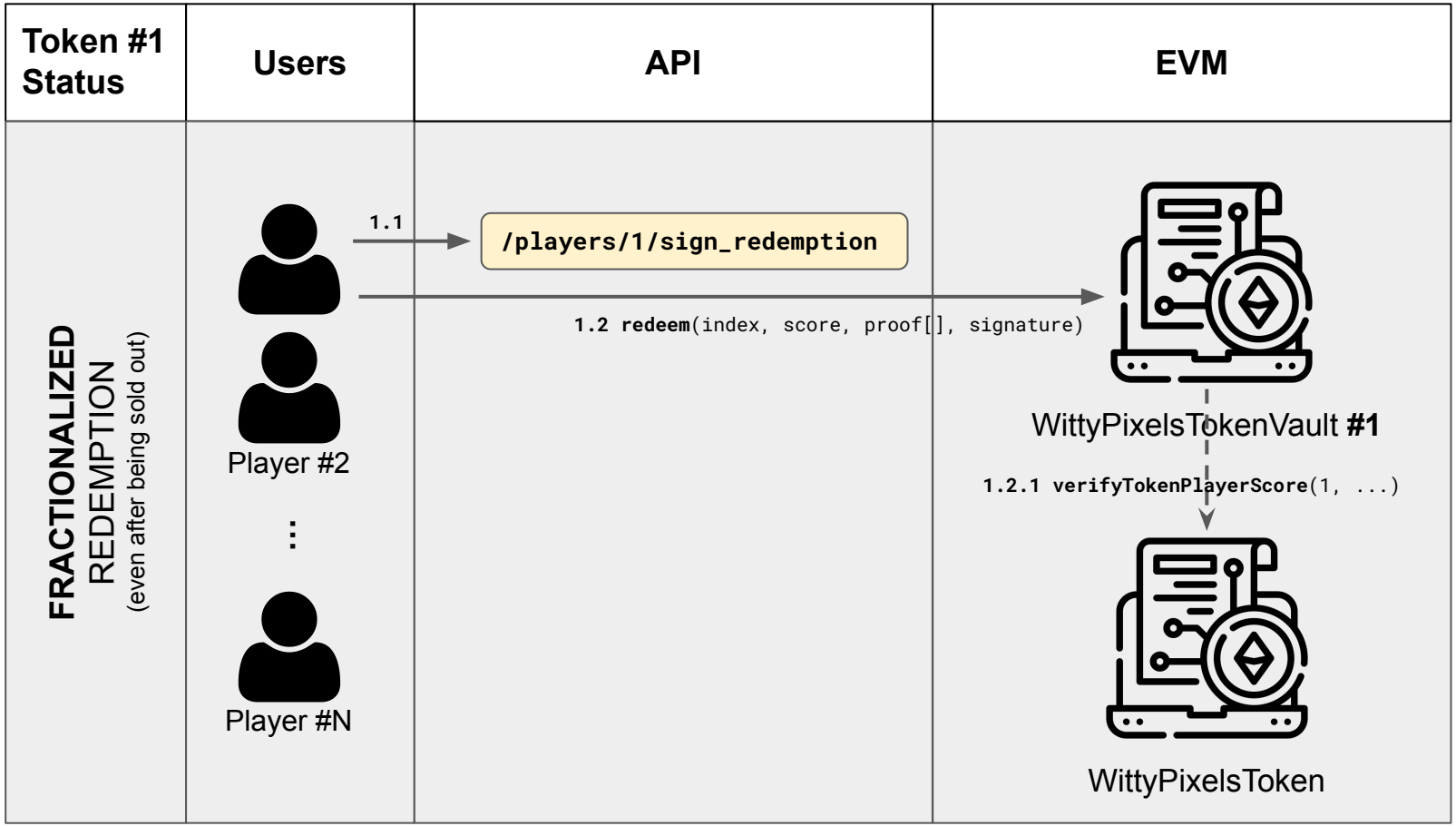
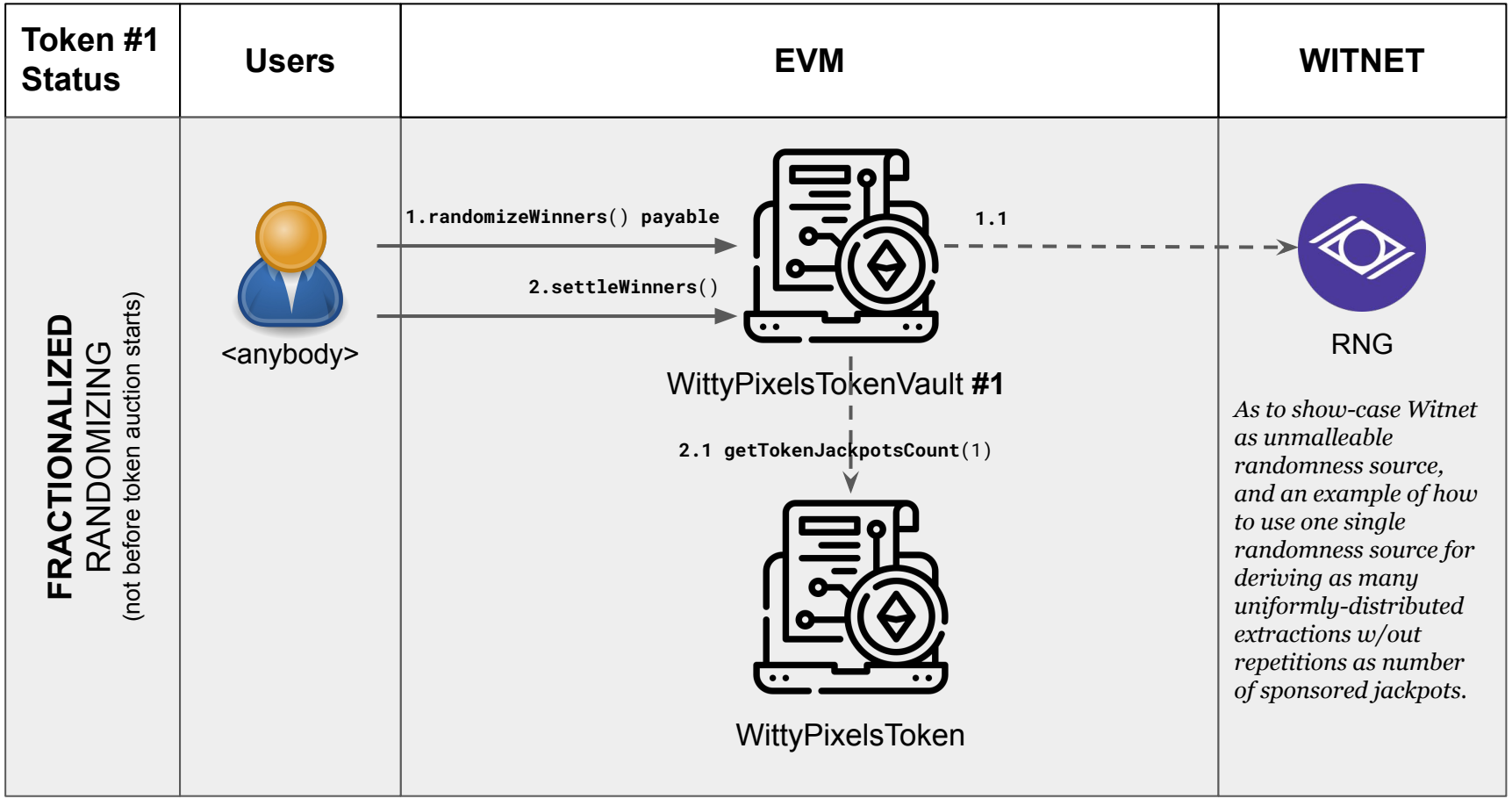

Token #1 Status
Users
EVM
VOID
WF
(deploy and verify)
launchNext(the_Event)
onlyOwner
returns (token_id)
WittyPixelsToken
<< upgradable >>
WittyPixelsTokenVault
<< clonable >>

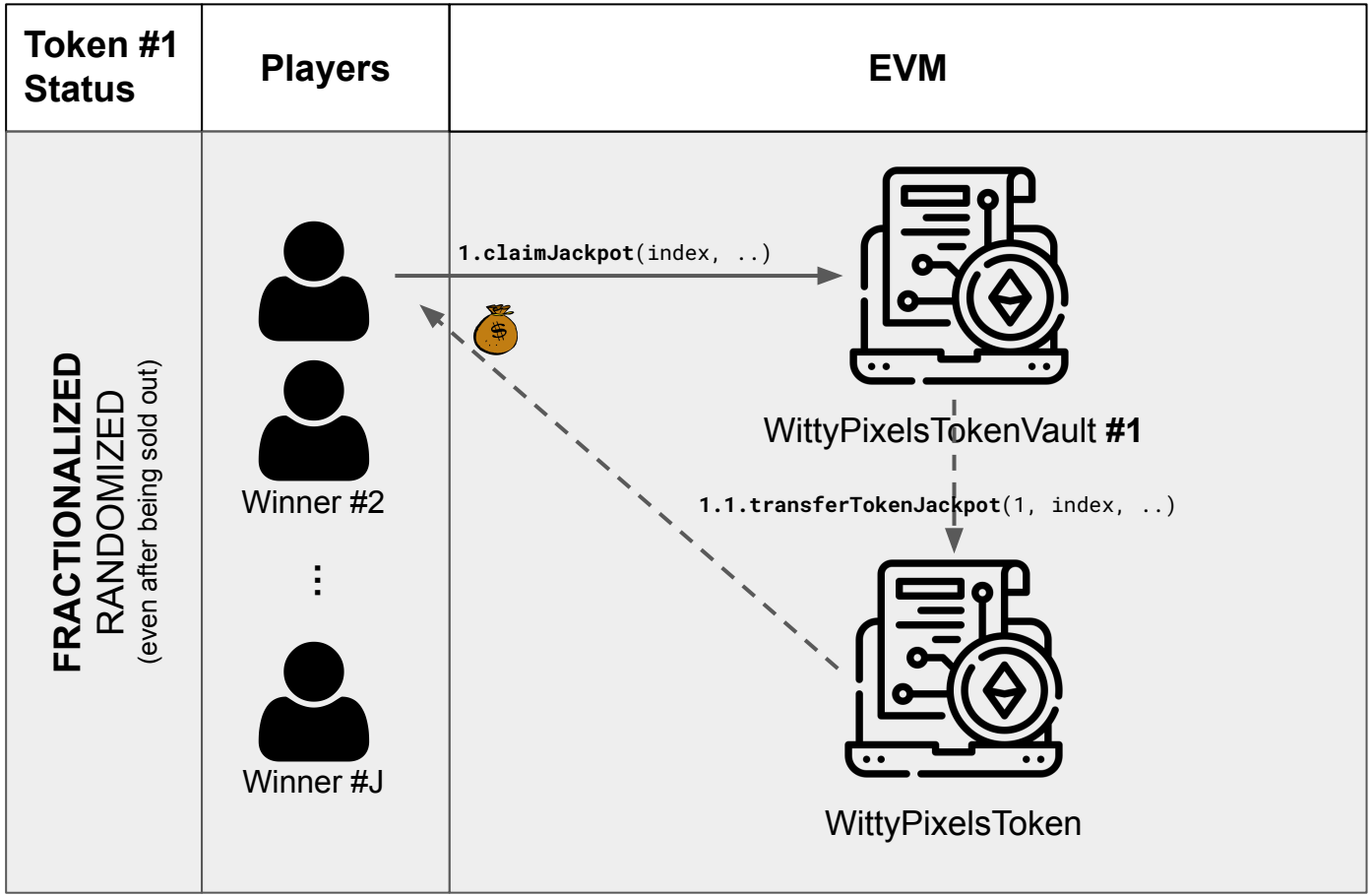

Token #1 Status
Users
EVM
LAUNCHING
(before token gets minted)
WF
Sponsors
1. setTokenSponsors(token_id, [{addr, text}]) onlyOwner
WittyPixelsToken
WittyPixelsTokenVault Prototype
2. sponsoriseToken(token_id) payable onlyTokenSponsors

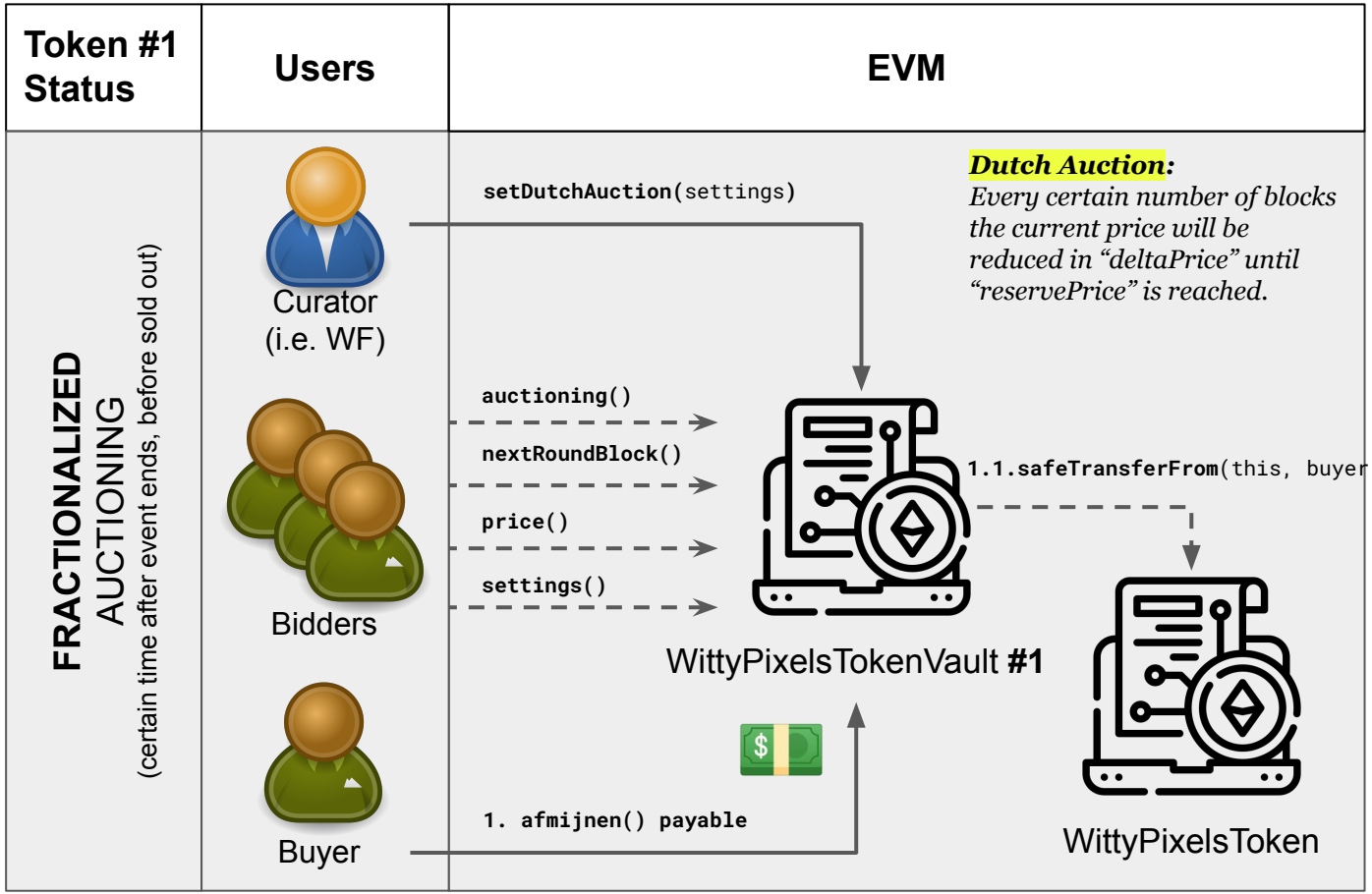
Token #1 Status	Users	API	WITNET	Users
LAUNCHING - after event starts - before event ends	 Player #2 ⋮ Player #N	`/leaderboard/1/` `/players/1/…`	 **HTTP/GET** *As to provide an auditable history of how the leaderboard evolved during the gameplay (as seen by players).*	 WF

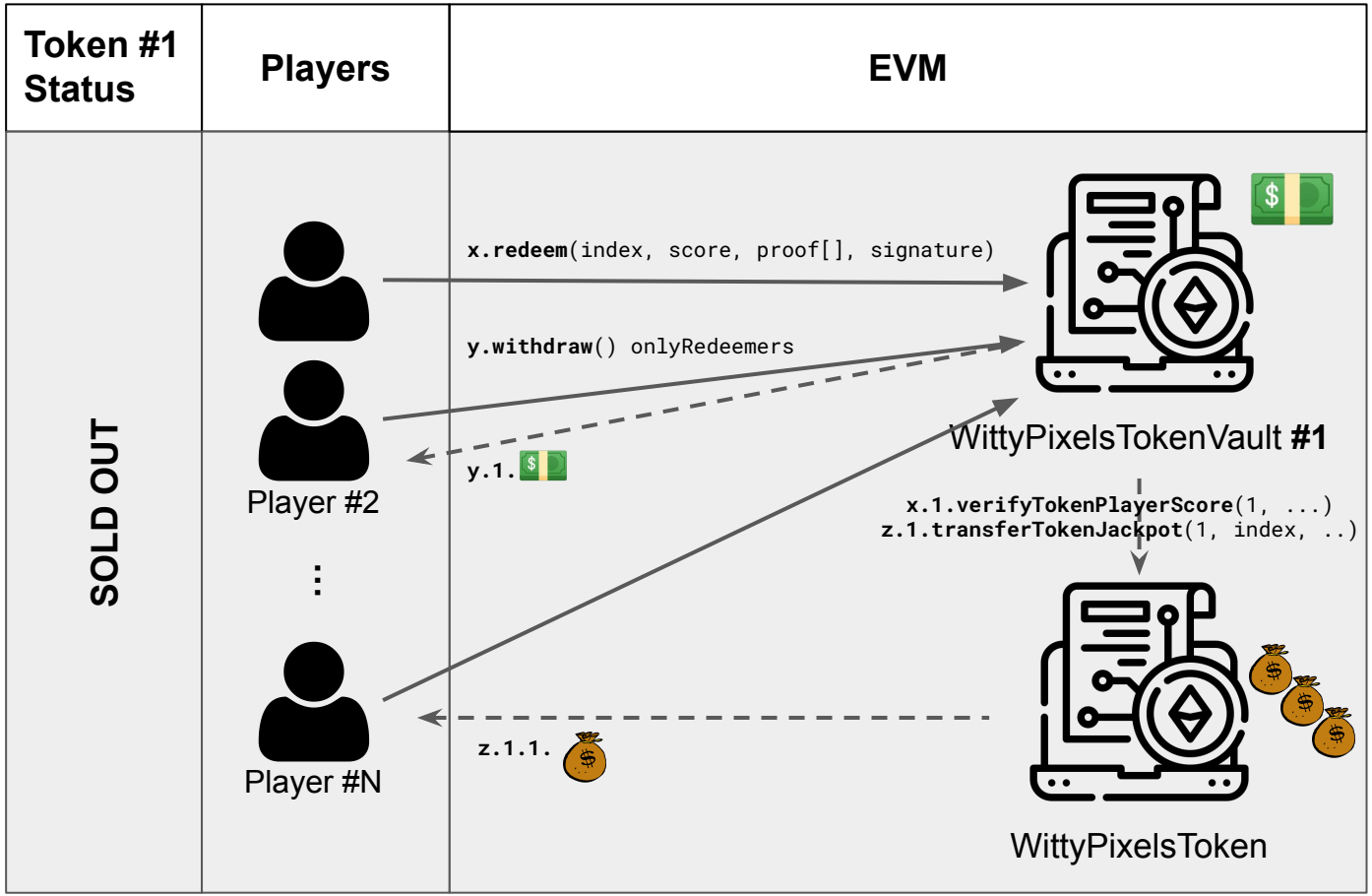

Token #1 Status
Users
EVM
WITNET
API
MINTING
(after event ends)
WF
1.premint(..)
2.fractionalize(..)
WittyPixelsToken
1.1
1.2
2.1
HTTP/HEAD
HTTP/GET
/image/1/
/leaderboard/1/
WittyPixelsTokenVault #1
As to ensure that (1) the token image actually exists upon minting, (2) any eventual change on the image content would be detected, and (3) published score distribution upon minting cannot be corrupted..


Token #1 Status
Users
API
EVM
FRACTIONALIZED
REDEMPTION
(even after being sold out)
1.1
/players/1/sign_redemption
1.2 redeem(index, score, proof[], signature)
Player #2
Player #N
WittyPixelsTokenVault #1
1.2.1 verifyTokenPlayerScore(1, ...)
WittyPixelsToken


Token #1 Status
Users
EVM
WITNET
FRACTIONALIZED
RANDOMIZING
(not before token auction starts)
<anybody>
1.randomizeWinners() payable
2.settleWinners()
1.1
WittyPixelsTokenVault #1
2.1 getTokenJackpotsCount(1)
WittyPixelsToken
RNG
As to show-case Witnet as unmalleable randomness source, and an example of how to use one single randomness source for deriving as many uniformly-distributed extractions w/out repetitions as number of sponsored jackpots.


Token #1 Status
Players
EVM
FRACTIONALIZED
RANDOMIZED
(even after being sold out)
1.claimJackpot(index, ..)
WittyPixelsTokenVault #1
1.1.transferTokenJackpot(1, index, ..)
Winner #2
Winner #J
WittyPixelsToken


Token #1 Status
Users
EVM
FRACTIONALIZED AUCTIONING
(certain time after event ends, before sold out)
Curator (i.e. WF)
setDutchAuction(settings)
Dutch Auction:
Every certain number of blocks the current price will be reduced in "deltaPrice" until "reservePrice" is reached.
auctioning()
nextRoundBlock()
price()
settings()
Bidders
WittyPixelsTokenVault #1
1.1.safeTransferFrom(this, buyer, 1)
WittyPixelsToken
1. afmijnen() payable
Buyer



Late redeemers or jackpot claims will be accepted even after the token gets sold out.