ERC-STANDARDS

Blockchain technologies, **lecture 5**

Course overview

- ERCs introduction
- ERC20
- ERC721
- ERC165
- ERC2612
- ERC1155
- ERC4626

ERC165

ERC165

- Creates a standard method to publish and detect what interfaces a smart contract implements.
- For some “standard interfaces” like the ERC-20 token interface, it is sometimes useful to query whether a contract supports the interface and if yes, which version of the interface.

ERC165

ERC2612

- EIP2612 extends the EIP-20 standard with a new function **permit**, which allows users to modify the mapping allowance using a signed message, instead of through msg.sender.
- Replay attacks are prevented using nonces.
- The owner can limit the time a Permit is valid for by setting deadline.
- Must implement:
  - function permit(address owner, address spender, uint value, uint deadline, uint8 v, bytes32 r, bytes32 s) external
  - function nonces(address owner) external view returns (uint)
  - function DOMAIN_SEPARATOR() external view returns (bytes32)

ERC2612

- Security concerns:
  - Front-running transactions: call permit before the intended party.
  - The relaying party can always choose to not submit the Permit after having received it.
  - owner != address(0) to avoid permit from creating an approval to spend “zombie funds” belong to the zero address.

ERC2612 – race condition

- Security concerns:
  - code depends on the order of the transactions.
  - a person who is running an Ethereum node can tell which transactions are going to occur before they are finalized.
  - example: contract sending rewards. Rewards in games, rewards as incentives in consensus protocols (reward finders of bad behavior).
  - In ERC20 tokens:
    - Alice -> approve(Eve, m)
    - Eve -> transfer(m) -- Eve sees the transaction Alice -> approve(Eve, n) before the transaction is included in blockchain.
    - Alice -> approve(Eve, n)
  - Mitigations: check expected value or setting approvals to 0 before changing them.

ERC1155

ERC1155

- EIP1155 standard interface for contracts that manage multiple token types. A single deployed contract may include any combination of fungible tokens, non-fungible tokens or other configurations (e.g. semi-fungible tokens).

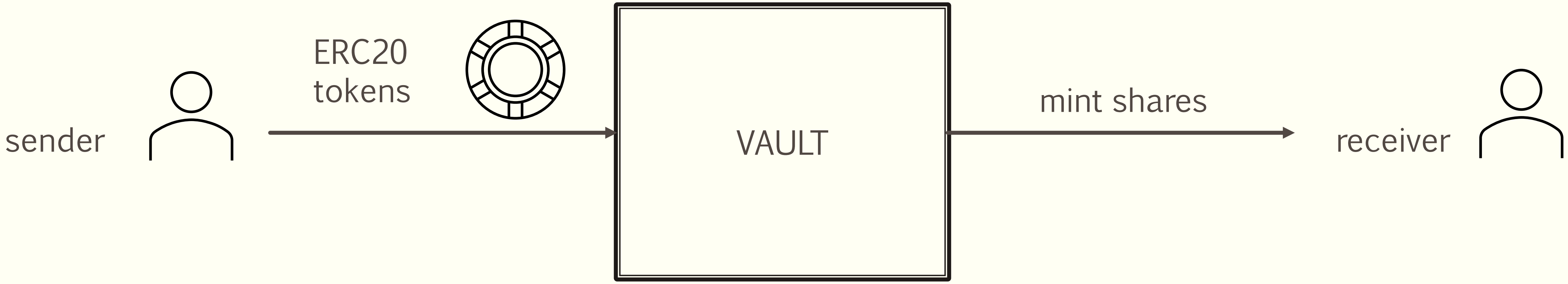
ERC4626

ERC4626

- Yield-bearing vaults – provide added gain on top of ordinary asset holding.
- Key component of decentralized finance (DeFi) platforms.
- ERC20 extension, offers basic functionality for depositing, withdrawing tokens and reading balances.
- Users deposit their tokens (ERC-20 tokens) into the vault, and in return, they receive vault-specific tokens (shares).
- Definitions:
  - **asset**: The underlying token managed by the Vault, EIP-20 contract.
  - **share**: The token of the Vault. Has a ratio of underlying assets exchanged on mint/deposit/withdraw/redeem (as defined by the Vault).
  - **fee**: An amount of assets or shares charged to the user by the Vault.

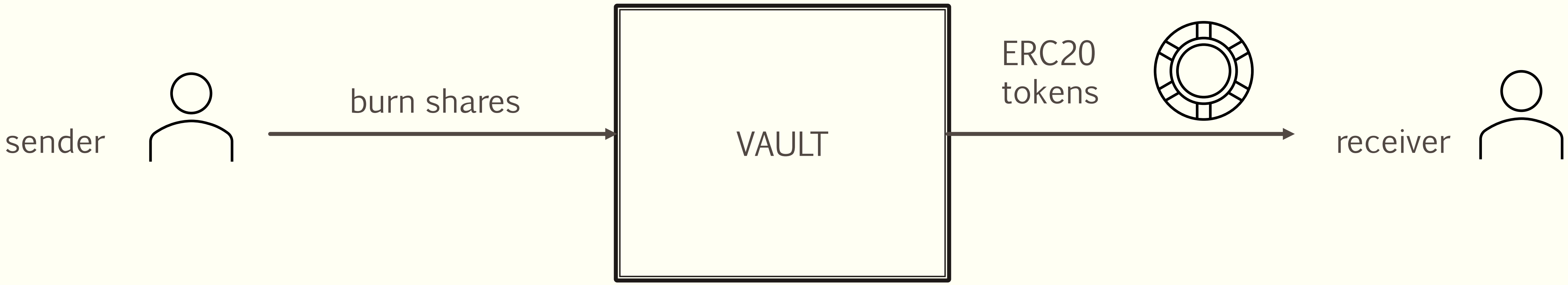
ERC4626

- EVENTS:
  - DEPOSIT
    - sender cand deposit assets into to vault.
    - in return vault mints shares.
    - shares are granted to a receiver.
    - sender can use both assets or shares to deposit tokens,
    - functions that emit deposit may receive as arguments both assets or shares to deposit tokens and apply conversion rates accordingly.



ERC4626

- EVENTS:
  - WITHDRAW
    - owner cand withdraw assets from the vault.
    - in return vault burns shares.
    - assets are transferred to a receiver.
    - sender can use both assets or shares to withdraw tokens,
    - functions that emit withdraw may receive as arguments both assets or shares to release tokens and burn shares after applying conversion rates accordingly.



ERC4626

- Functions:
  - function asset() public view returns (address)

    Address of the underlying ERC20 token.
  - function totalAssets() public view returns (uint256)

    Total amount of underlying assets held by the vault.
  - function convertToShares(uint256 assets) public view returns (uint256 shares)
  - function convertToAssets(uint256 shares) public view returns (uint256 assets)

    Returns the amount of shares/assets that would be exchanged for the amount of assets/shares provided.
  - function deposit(uint256 assets, address receiver) public returns (uint256 shares)

    Deposits assets of underlying tokens into the vault and grants ownership of shares to receiver.
  - function mint(uint256 shares, address receiver) public returns (uint256 assets)

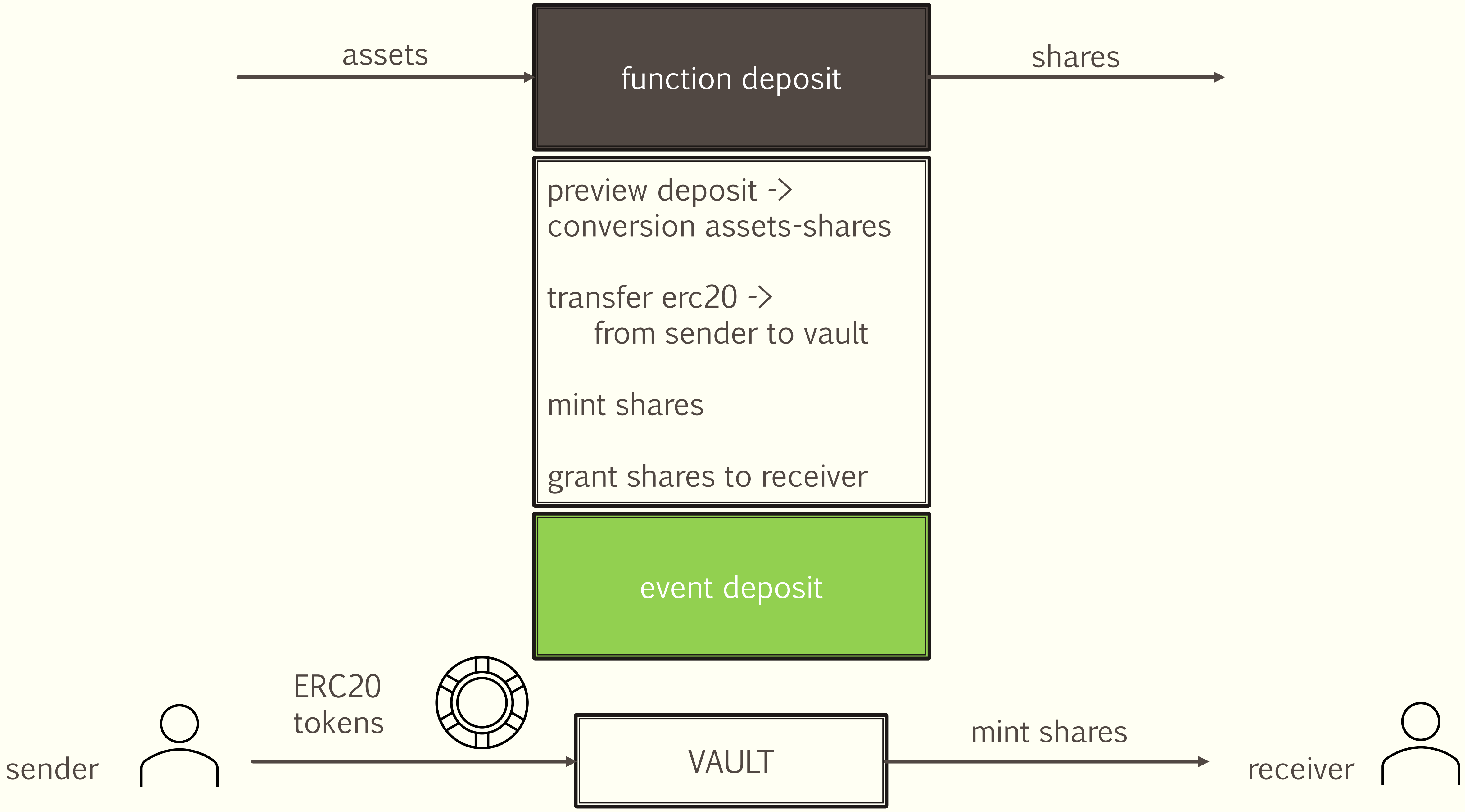
    Mints exactly shares vault shares to receiver by depositing assets of underlying tokens.
  - function withdraw(uint256 assets, address receiver, address owner) public returns (uint256 shares)

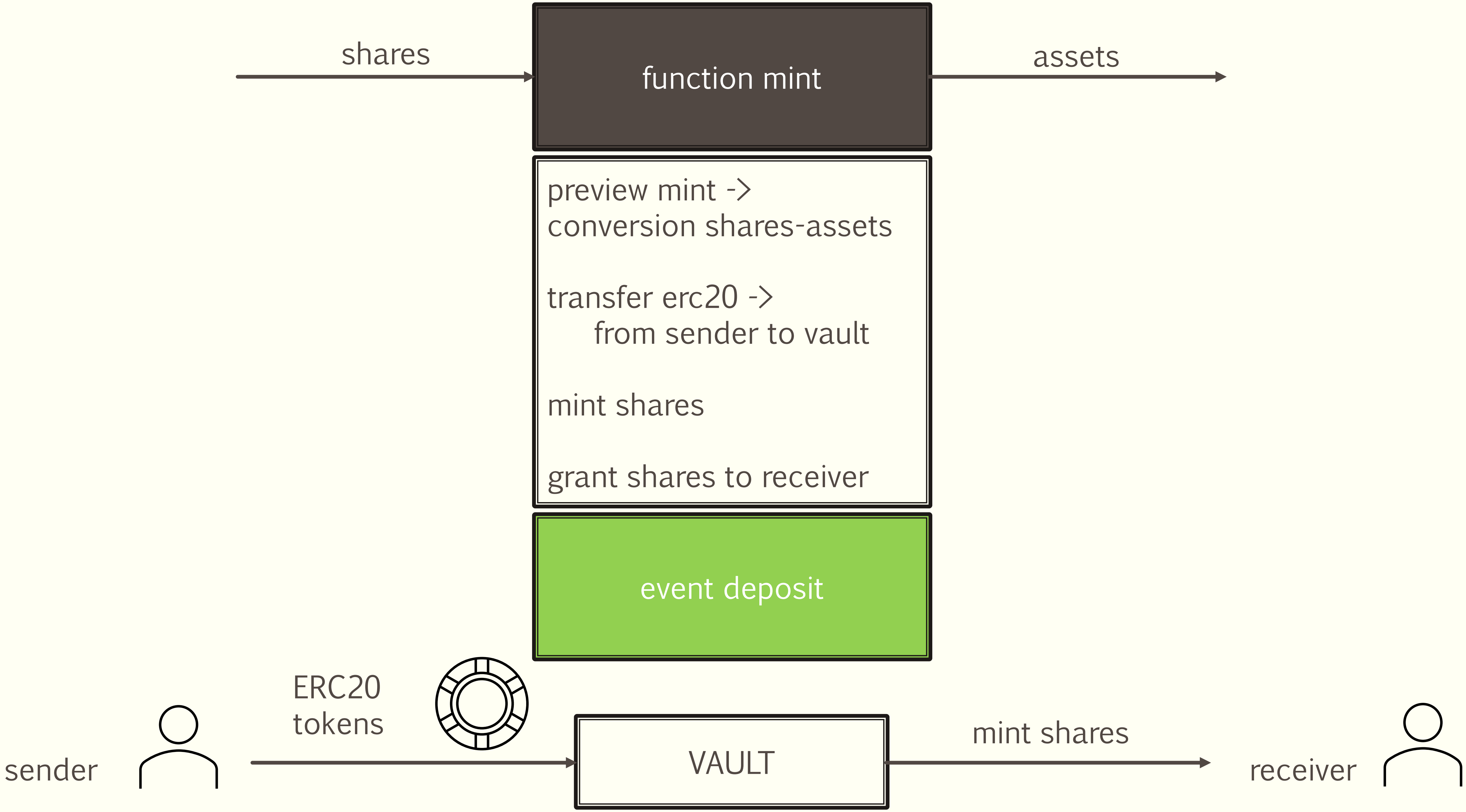
    Burns shares from owner and send exactly assets token from the vault to receiver.
  - function redeem(uint256 shares, address receiver, address owner) public returns (uint256 assets)

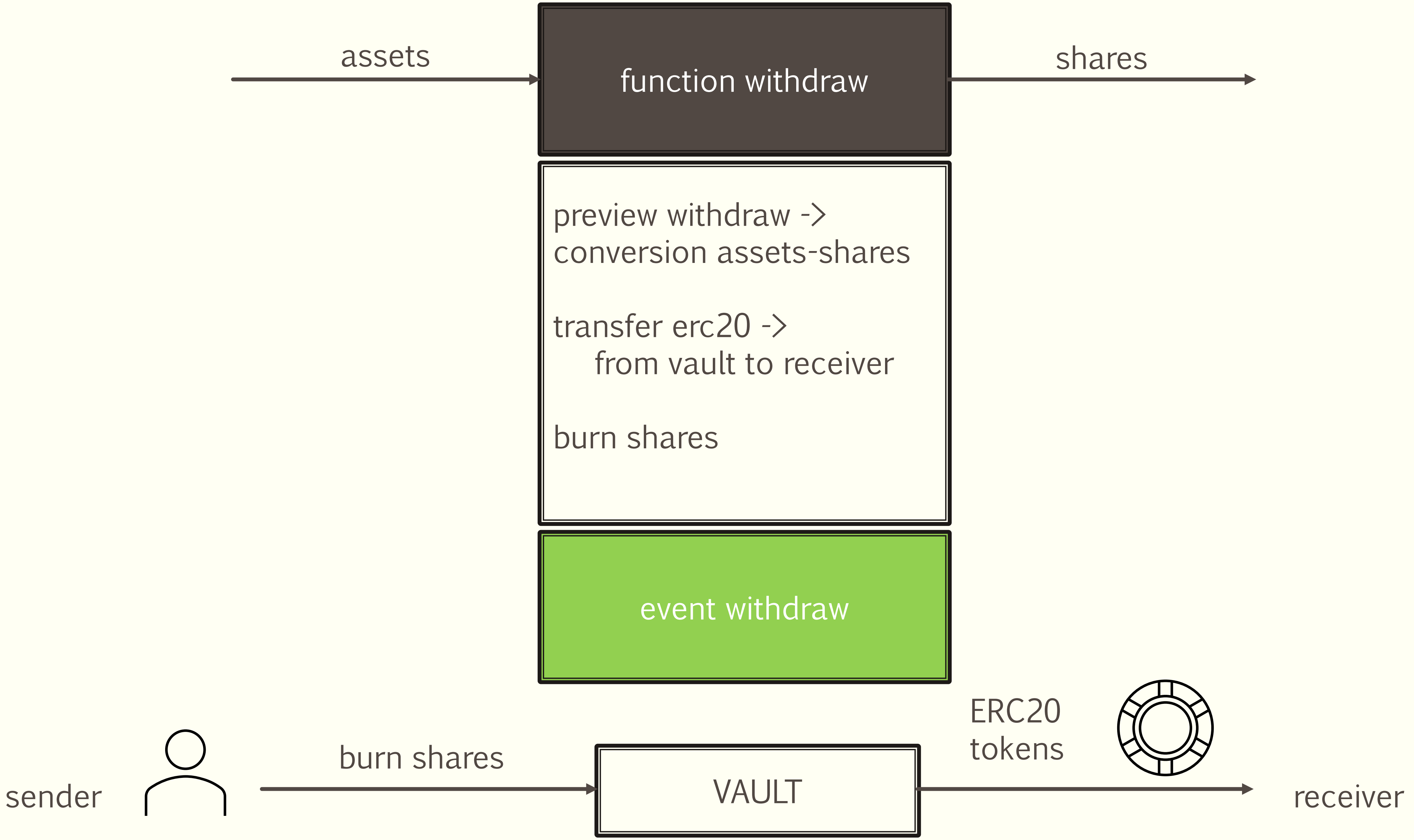
    Redeems a specific number of shares from owner and sends assets of underlying token from the vault to receiver.

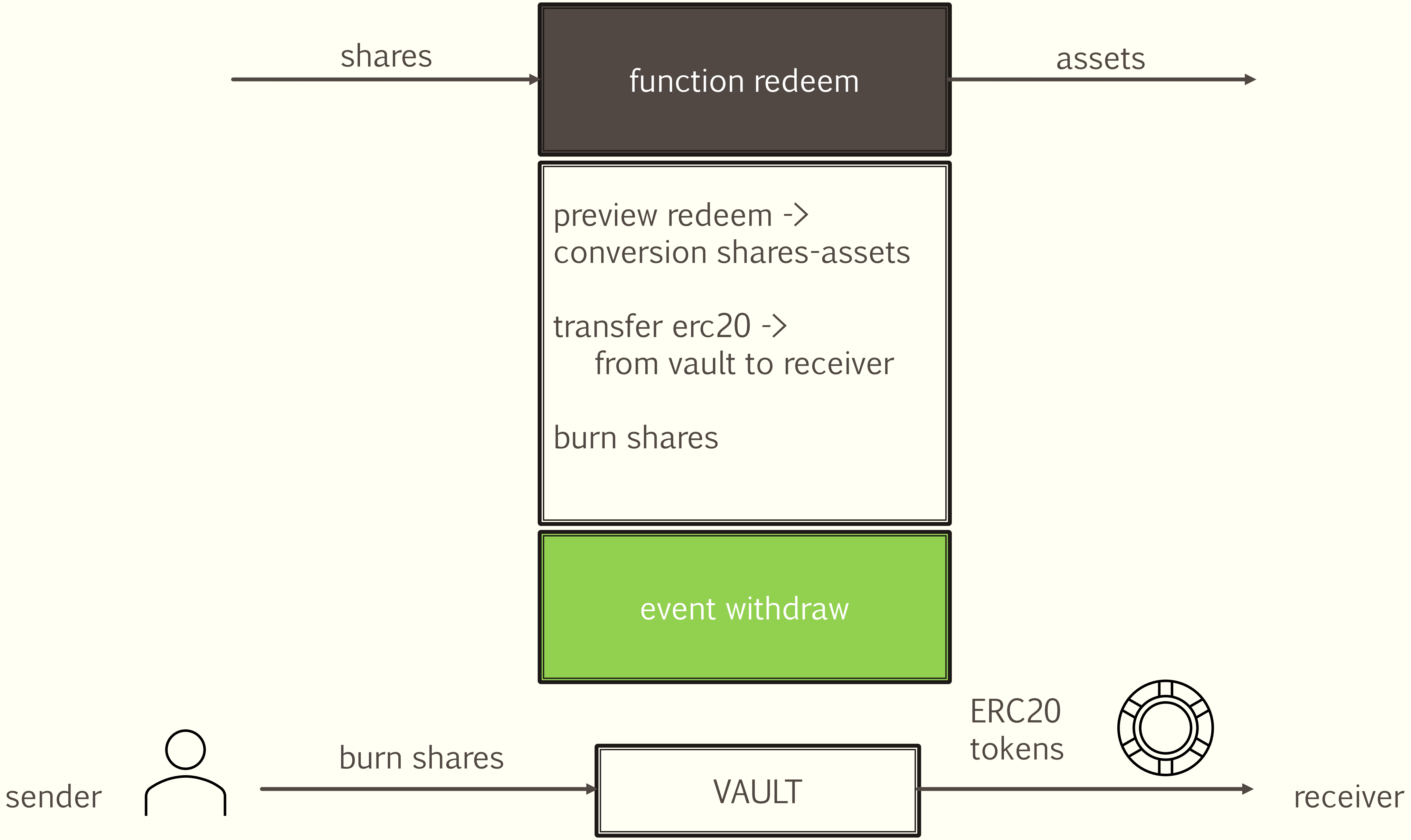
ERC4626

- Other functions:
  - previewDeposit, maxDeposit;
  - previewMint, maxMint;
  - previewRedeem, maxRedeem;
  - previewWithdraw, maxWithdraw;
  - totalSupply, balanceOf   -- refer to sheres.


assets
function deposit
shares
preview deposit ->
conversion assets-shares
transfer erc20 ->
from sender to vault
mint shares
grant shares to receiver
event deposit
ERC20
tokens
sender
VAULT
mint shares
receiver


shares
function mint
assets
preview mint ->
conversion shares-assets
transfer erc20 ->
from sender to vault
mint shares
grant shares to receiver
event deposit
ERC20
tokens
sender
VAULT
mint shares
receiver


assets
function withdraw
shares
preview withdraw ->
conversion assets-shares
transfer erc20 ->
from vault to receiver
burn shares
event withdraw
sender
burn shares
VAULT
ERC20
tokens
receiver


shares
function redeem
assets
preview redeem ->
conversion shares-assets
transfer erc20 ->
from vault to receiver
burn shares
event withdraw
sender
burn shares
VAULT
ERC20
tokens
receiver

Bibliography

- EIP615 https://eips.ethereum.org/EIPS/eip-165
- EIP2612 https://eips.ethereum.org/EIPS/eip-2612
- Race condition https://swcregistry.io/docs/SWC-114/
- ERC4626 https://ethereum.org/developers/docs/standards/tokens/erc-4626
- ERC-4626 https://www.quicknode.com/guides/ethereum-development/smart-contracts/how-to-use-erc-4626-with-your-smart-contract
- ERC-4626 implementation https://github.com/transmissions11/solmate/blob/main/src/tokens/ERC4626.sol
- https://www.quicknode.com/guides/ethereum-development/nfts/how-to-create-and-deploy-an-erc-721-nft
- https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol