Avoiding Token Loss

warning Careful! Losing tokens means losing money! Token loss is possible under multiple scenarios. These scenarios can be grouped into a few related classes:

1. Improper key management
2. Refunding deleted accounts
3. Failed function calls in batches

Improper key management

Improper key management may lead to token loss. Mitigating such scenarios may be done by issuing backup keys allowing for recovery of accounts whose keys have been lost or deleted.

Loss ofFullAccess

key

A user may lose their private key of aFullAccess key pair for an account with no other keys. No one will be able to recover the funds. Funds will remain locked in the account forever.

Loss ofFunctionCall

access key

An account may have its one and onlyFunctionCall access key deleted. No one will be able to recover the funds. Funds will remain locked in the account forever.

Refunding deleted accounts

When a refund receipt is issued for an account, if that account no longer exists, the funds will be dispersed among validators proportional to their stake in the current epoch.

Deleting account with non-existent beneficiary

When you delete an account, you must assign a beneficiary. Once deleted, a transfer receipt is generated and sent to the beneficiary account. If the beneficiary account does not exist, a refund receipt will be generated and sent back to the original account. Since the original account has already been deleted, the funds will be dispersed among validators.

Account with zero balance is garbage-collected, just before it receives refund

If an accountA transfers all of its funds to another accountB and accountB does not exist, a refund receipt will be generated for accountA . During the period of this round trip, accountA is vulnerable to deletion by garbage collection activities on the network. If accountA is deleted before the refund receipt arrives, the funds will be dispersed among validators.

Failed function calls in batches

warning When designing a smart contract, you should always consider the asynchronous nature of NEAR Protocol. If a contract functionf1 calls two (or more) other functionsf2 andf3 , and at least one of these functions,f2 andf3 fails, then tokens will be refunded from the function that failed, but tokens will be appropriately credited to the function(s) which succeed.

The successful call's tokens may be considered lost depending on your use case if a single failure in the batch means the whole batch failed. Edit this page Last updatedonDec 11, 2023 byDamian Parrino Was this page helpful? Yes No

Previous Tokens Next Storage on NEAR