Native - from v1.11

Optionally perform state migrations of Secret Network smart contracts

Contract Migration on Secret Network

Contracts on Secret Network can be initialized asmigratable , which allows the contract administrator to upload a new version of the contract and then send amigrate message to move to the new code, which "migrates" storage from a previous contract to the new contract.

There are two key components to a migratable contract:

- A designatedadmin
- whose address will be allowed to perform migrations.
- The availability of aMigrateMsg
- transaction.
-

Contract Admin

Oninstantiation , the contract creator can specify anadmin address, which can be the contract creator's wallet address, an external account, or a governance contract, etc. Whatever the address is, thisadmin gains the ability to migrate the contract to a newcodeId andcodeHash , and can also update or clear theadmin address. When theadmin invokes theMigrateMsg message, themigrate() function is called on the new contract, where the new contract can optionally perform state migrations from an old contract.

Once themigrate() function is invoked, the contract address now points to the new code, and anybody contacting that address will reach the new code. The old code becomes unreachable.

MigrateMsg

Performing a contract migration is a three step process:

1. Write a newer version of the contract you wish to update
2. Upload the new smart contract, but don’t instantiate it
3. Use a dedicatedMigrateMsg
4. transaction to point the new contract to the code you wish to migrate
5.

When migrating, the new contract must have amigrate function as anentry_point.

```

Copy

# [entry_point]

pubfnmigrate(_deps:DepsMut, _env:Env, msg:MigrateMsg)->StdResult { matchmsg { MigrateMsg::Migrate{}=>Ok(Response::default()), MigrateMsg::StdError{}=>Err(StdError::generic_err("this is an std error")), } }

```

Themigrate function provides the ability to make any desired changes to the contract's state, similar to a database migration.

If themigrate function returns an error, the transaction will abort and no state changes will occur.

Execute Migration with Secret.js

```

Copy import{ SecretNetworkClient,Wallet }from"secretjs"; importdotenvfrom"dotenv"; dotenv.config();

constwallet=newWallet(process.env.MNEMONIC);

constsecretjs=newSecretNetworkClient({ chainId:"pulsar-3", url:"https://api.pulsar.scrttestnet.com", wallet:wallet, walletAddress:wallet.address, });

constcodeId=1;// codeId for new contract constcontractCodeHash="";// codeHash for new contract
```

constcontractAddress="";// contract address, which doesn't change upon migration

letmain=async()=>{ constmigrateMsg={ remove_users:{ a,b },// this is an example. migrateMsg can be left empty if contract storage is unchanged during migration };

consttx=awaitsecretjs.tx.compute.migrateContract({ code_id:codeId, contract_address:contractAddress, sender:wallet.address, code_hash:contractCodeHash, msg:migrateMsg, }, { gasLimit:400_000, });

console.log(tx.rawLog); };

main();

```

Query contract migratability

Secret contracts instantiated prior to the v1.11 network upgrade can be migrated via a SCRT governance signaling proposal. If approved for migration, the contracts' code hashes will be matched to a chosen admin key and added to a list which is hardcoded inside the enclave for reference.

Further reading on hardcoded admin list In order for a Secret contract instantiated after the v1.11 network upgrade to be migrated,the contract must be instantiated with an admin . To query whether or not a Secret contract was instantiated with anadmin , use thecontractInfo method:

```

Copy letqueryContractInfo=async()=>{ letquery=awaitsecretjs.query.compute.contractInfo({ contract_address:contractAddress, code_hash:contractCodeHash, });

console.log(query); }; queryContractInfo();

```

The query will return the admin address, if there is one:

```

Copy { contract_address:'secret15l8cqadh5pruweuxvr5ku830hamqny8ey2q2vf', contract_info:{ code_id:'1303', creator:'secret1j7n3xx4sfgjea4unghd78qvnvxdz49cxmrkqlj', label:'migrate example', created:{ block_height:'1088301',tx_index:'0'}, ibc_port_id:'', admin:'secret1j7n3xx4sfgjea4unghd78qvnvxdz49cxmrkqlj', admin_proof:'TSxuJZZG0/eYGggmXNXw79So9jET3zLIy2An9bB5dA0=' } }

```

To query if a Secret contract was migrated successfully , use thecontractHistory method:

```

Copy letqueryContractHistory=async()=>{ letquery=awaitsecretjs.query.compute.contractHistory({ contract_address:contractAddress, code_hash:contractCodeHash, });

console.log(query); }; queryContractHistory();

```

The method is designed to retrieve the history of a contract, specifically its code changes over time. The method returns an object containing an array ofContractCodeHistoryEntry items.

Last updated6 months ago On this page *Contract Migration on Secret Network * Contract Admin * MigrateMsg * Execute Migration with Secret.js * Query contract migratability

Was this helpful? Edit on GitHub Export as PDF
```