Sending Messages

Learn how to send messages with SecretJS.

Secret Network Client Setup

```
Copy import{ SecretNetworkClient,Wallet }from"secretjs";

constwallet=newWallet("Your mnemonic words go here");

constsecretjs=newSecretNetworkClient({ chainId:"pulsar-3", url:"https://api.pulsar3.scrttestnet.com", wallet:wallet,
walletAddress:wallet.address, });
```

SecretJS Messages

On a signer secret.js,secretjs.tx is used to broadcast transactions. Every function undersecretjs.tx can receive an optionalTxOptions .

Full API »

secretjs.tx.broadcast()

Used to send a complex transactions, which contains a list of messages. The messages are executed in sequence, and the transaction succeeds if all messages succeed.

For a list of all messages see https://secretjs.scrt.network/interfaces/Msg

```
Copy constaddMinterMsg=newMsgExecuteContract({ sender:MY_ADDRESS, contract_address:MY_NFT_CONTRACT,
code_hash:MY_NFT_CONTRACT_CODE_HASH,// optional but way faster msg:{ add_minters:{ minters:[MY_ADDRESS] } },
sent_funds:[],// optional });

constmintMsg=newMsgExecuteContract({ sender:MY_ADDRESS, contract_address:MY_NFT_CONTRACT,
code_hash:MY_NFT_CONTRACT_CODE_HASH,// optional but way faster msg:{ mint_nft:{ token_id:"1",
owner:MY_ADDRESS, public_metadata:{ extension:{ image:"https://scrt.network/secretnetwork-logo-secondary-black.png",
name:"secretnetwork-logo-secondary-black", }, }, private_metadata:{ extension:{ image:"https://scrt.network/secretnetwork-
logo-primary-white.png", name:"secretnetwork-logo-primary-white", }, }, }, }, sent_funds:[],// optional });

consttx=awaitsecretjs.tx.broadcast([addMinterMsg,mintMsg],{ gasLimit:200_000, });
```

secretjs.tx.simulate()

Used to simulate a complex transactions, which contains a list of messages, without broadcasting it to the chain. Can be used to get a gas estimation or to see the output without actually committing a transaction on-chain.

The input should be exactly how you'd use it insecretjs.tx.broadcast() , except that you don't have to pass ingasLimit ,gasPriceInFeeDenom &feeDenom .

Notes:

- On mainnet, it's recommended to not simulate every transaction as this can burden your node provider. Instead, use this while testing to determine the gas limit for each of your app's transactions, then in production use hard-coded values.
- Gas estimation is known to be a bit off, so you might need to adjust it a bit before broadcasting.
- MsgInstantiateContract
- ,MsgExecuteContract
- ,MsgMigrateContract
- ,MsgUpdateAdmin
- &MsgClearAdmin
- simulations are not supported for security reasons.
-

```
```

Copy constsendToAlice=newMsgSend({ from_address:bob, to_address:alice, amount:stringToCoins("1uscrt"), });

constsendToEve=newMsgSend({ from_address:bob, to_address:eve, amount:stringToCoins("1uscrt"), });

constsim=awaitsecretjs.tx.simulate([sendToAlice,sendToEve]);

consttx=awaitsecretjs.tx.broadcast([sendToAlice,sendToEve],{ // Adjust gasLimit up by 10% to account for gas estimation error gasLimit:Math.ceil(sim.gasInfo.gasUsed*1.1), });

```

secretjs.tx.signTx()

Used to sign transactions independently from the broadcast process. This is useful when you want to keep your seed safe and sign transactions offline.

secretjs.tx.broadcastSignedTx()

Used to send offline signed transactions.

```

Copy constbob="secret1dgqnta7fwjj6x9kusyz7n8vpl73l7wsm0gaamk"; constmsg=newMsgSend({ from_address:myAddress, to_address:bob, amount:stringToCoins("1000000uscrt"), });

letsignedTX=awaitsecretjs.tx.signTx([msg],{ gasLimit:20_000, gasPriceInFeeDenom:0.1, feeDenom:"uscrt", });

lettx=awaitsecretjs.tx.broadcastSignedTx(signedTX);

```

secretjs.tx.authz.exec()

MsgExec attempts to execute the provided messages using authorizations granted to the grantee. Each message should have only one signer corresponding to the granter of the authorization.

Input:MsgExecParams

secretjs.tx.authz.exec.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.authz.grant()

MsgGrant is a request type for Grant method. It declares authorization to the grantee on behalf of the granter with the provided expiration time.

Input:MsgGrantParams

secretjs.tx.authz.grant.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.authz.revoke()

MsgRevoke revokes any authorization with the provided sdk.Msg type on the granter's account with that has been granted to the grantee.

Input:MsgRevokeParams

secretjs.tx.authz.revoke.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.bank.multiSend()

MsgMultiSend represents an arbitrary multi-in, multi-out send message.

Input:MsgMultiSendParams
```

```
Copy consttx=awaitsecretjs.tx.bank.multiSend( { inputs:[ { address:myAddress, coins:stringToCoins("2uscrt"), }, ], outputs:[ { address:alice, coins:stringToCoins("1uscrt"), }, { address:bob, coins:stringToCoins("1uscrt"), }, ], }, { gasLimit:20_000, }, );
```

```
Copy

secretjs.tx.bank.multiSend.simulate()
```

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.bank.send()

MsgSend represents a message to send coins from one account to another.

Input:MsgSendParams

```
Copy consttx=awaitsecretjs.tx.bank.send( { from_address:myAddress, to_address:alice, amount:stringToCoins("1uscrt"), }, { gasLimit:20_000, }, );
```

secretjs.tx.bank.send.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.compute.storeCode()

Upload a compiled contract to Secret Network

Input:MsgStoreCodeParams

```
Copy consttx=awaitsecretjs.tx.compute.storeCode( { sender:myAddress, wasm_byte_code:fs.readFileSync( {__dirname}/snip20-ibc.wasm.gz, )asUint8Array, source:"", builder:"", }, { gasLimit:1_000_000, }, );

constcodeId=Number( tx.arrayLog.find((log)=>log.type==="message"&&log.key==="code_id") .value, );
```

secretjs.tx.compute.storeCode.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.compute.instantiateContract()

Instantiate a contract from code id

Input: [MsgInstantiateContractParams](https://secretjs.scrt.network/interfaces/MsgInstanti

ateContractParams)

```
Copy consttx=awaitsecretjs.tx.compute.instantiateContract( { sender:myAddress, admin:myAddress,// optional admin address that can perform code migrations code_id:codeId, code_hash:codeHash,// optional but way faster initMsg:{ name:"Secret SCRT", admin:myAddress, symbol:"SSCRT", decimals:6, initial_balances:[{ address:myAddress,amount:"1"}], prng_seed:"eW8=", config:{ public_total_supply:true, enable_deposit:true, enable_redeem:true, enable_mint:false, enable_burn:false, }, supported_denoms:["uscrt"], }, label:"sSCRT", init_funds:[],// optional }, { gasLimit:100_000, }, );

constcontractAddress=tx.arrayLog.find( (log)=>log.type==="message"&&log.key==="contract_address", ).value;
```

```

secretjs.tx.compute.instantiateContract.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

WARNING:secretjs.tx.compute simulations are not supported for security reasons.

secretjs.tx.compute.executeContract()

Execute a function on a contract

Input:MsgExecuteContractParams

```

Copy consttx=awaitsecretjs.tx.compute.executeContract({ sender:myAddress, contract_address:contractAddress, code_hash:codeHash,// optional but way faster msg:{ transfer:{ recipient:bob, amount:"1", }, }, sent_funds:[],// optional }, { gasLimit:100_000, },);

```

secretjs.tx.compute.executeContract.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

WARNING:secretjs.tx.compute simulations are not supported for security reasons.

secretjs.tx.compute.migrateContract()

Migrate a contract's code while keeping the same address. Invokes themigrate() function on the new code.

Input:MsgMigrateContractParams

```

Copy consttx=awaitsecretjs.tx.compute.migrateContract({ sender:myAddress, contract_address:contractAddress, code_id:newCodeId, code_hash:codeHash,// optional but way faster msg:{ migrate_state_to_new_format:{}, }, sent_funds:[],// optional }, { gasLimit:100_000, },);

```

secretjs.tx.compute.migrateContract.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

WARNING:secretjs.tx.compute simulations are not supported for security reasons.

secretjs.tx.compute.updateAdmin()

Update a contract's admin.

Input:MsgUpdateAdminParams

```

Copy consttx=awaitsecretjs.tx.compute.updateAdmin({ sender:currentAdminAddress, contract_address:contractAddress, new_admin:newAdminAddress, }, { gasLimit:100_000, },);

```

secretjs.tx.compute.updateAdmin.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

WARNING:secretjs.tx.compute simulations are not supported for security reasons.

secretjs.tx.compute.clearAdmin()
```

clear a contract's admin.

Input:MsgClearAdminParams

```

Copy consttx=awaitsecretjs.tx.compute.clearAdmin( { sender:currentAdminAddress, contract_address:contractAddress, }, { gasLimit:100_000, }, );

```

secretjs.tx.compute.clearAdmin.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

WARNING:secretjs.tx.compute simulations are not supported for security reasons.

secretjs.tx.crisis.verifyInvariant()

MsgVerifyInvariant represents a message to verify a particular invariance.

Input:MsgVerifyInvariantParams

secretjs.tx.crisis.verifyInvariant.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.distribution.fundCommunityPool()

MsgFundCommunityPool allows an account to directly fund the community pool.

Input:MsgFundCommunityPoolParams

```

Copy consttx=awaitsecretjs.tx.distribution.fundCommunityPool( { depositor:myAddress, amount:stringToCoins("1uscrt"), }, { gasLimit:20_000, }, );

```

secretjs.tx.distribution.fundCommunityPool.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.distribution.setWithdrawAddress()

MsgSetWithdrawAddress sets the withdraw address for a delegator (or validator self-delegation).

Input:MsgSetWithdrawAddressParams

```

Copy consttx=awaitsecretjs.tx.distribution.setWithdrawAddress( { delegator_address:mySelfDelegatorAddress, withdraw_address:myOtherAddress, }, { gasLimit:20_000, }, );

```

secretjs.tx.distribution.setWithdrawAddress.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.distribution.withdrawDelegatorReward()

MsgWithdrawDelegatorReward represents delegation withdrawal to a delegator from a single validator.

Input:MsgWithdrawDelegatorRewardParams

```

Copy consttx=awaitsecretjs.tx.distribution.withdrawDelegatorReward( { delegator_address:myAddress,
```

validator_address:someValidatorAddress, }, { gasLimit:20_000, },);

```

secretjs.tx.distribution.withdrawDelegatorReward.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.distribution.withdrawValidatorCommission()

MsgWithdrawValidatorCommission withdraws the full commission to the validator address.

Input:MsgWithdrawValidatorCommissionParams

```

Copy consttx=awaitsecretjs.tx.distribution.withdrawValidatorCommission({ validator_address:myValidatorAddress, }, { gasLimit:20_000, },);

```

Or a better one:

```

Copy consttx=awaitsecretjs.tx.broadcast([newMsgWithdrawDelegatorReward({ delegator_address:mySelfDelegatorAddress, validator_address:myValidatorAddress, }), newMsgWithdrawValidatorCommission({ validator_address:myValidatorAddress, }),], { gasLimit:30_000, },);

```

secretjs.tx.distribution.withdrawValidatorCommission.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.evidence.submitEvidence()

MsgSubmitEvidence represents a message that supports submitting arbitrary evidence of misbehavior such as equivocation or counterfactual signing.

Input:MsgSubmitEvidenceParams

secretjs.tx.evidence.submitEvidence.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.feegrant.grantAllowance()

MsgGrantAllowance adds permission for Grantee to spend up to Allowance of fees from the account of Granter.

Input:MsgGrantAllowanceParams

```

Copy constnewWallet=newWallet();

consttxGranter=awaitsecretjsGranter.tx.feegrant.grantAllowance({ granter:secretjsGranter.address, grantee:newWallet.address, allowance:{ spend_limit:stringToCoins("1000000uscrt"), }, });

constsecretjsGrantee=newSecretNetworkClient({ url:"http://localhost:1317", chainId:"secretdev-1", wallet:newWallet, walletAddress:newWallet.address, });

// Send a tx from newWallet with secretjs.address as the fee payer cosnt txGrantee=awaitsecretjsGrantee.tx.gov.submitProposal({ proposer:secretjsGrantee.address, type:ProposalType.TextProposal, initial_deposit:[], content:{ title:"Send a tx without any balance", description:Thanks{secretjsGranter.address}!, }, }, { feeGranter:secretjsGranter.address, },);

```

secretjs.tx.feegrant.grantAllowance.simulate()
```

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.feegrant.revokeAllowance()

MsgRevokeAllowance removes any existing Allowance from Granter to Grantee.

Input:MsgRevokeAllowanceParams

```

Copy consttx=awaitsecretjs.tx.feegrant.revokeAllowance({ granter:secretjs.address, grantee:newWallet.address, });

```

secretjs.tx.feegrant.revokeAllowance.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.gov.deposit()

MsgDeposit defines a message to submit a deposit to an existing proposal.

Input:MsgDepositParams

```

Copy consttx=awaitsecretjs.tx.gov.deposit( { depositor:myAddress, proposal_id:someProposalId, amount:stringToCoins("1uscrt"), }, { gasLimit:20_000, }, );

```

secretjs.tx.gov.deposit.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.gov.submitProposal()

MsgSubmitProposal defines an sdk.Msg type that supports submitting arbitrary proposal Content.

Input:MsgSubmitProposalParams

```

Copy consttx=awaitsecretjs.tx.gov.submitProposal( { type:ProposalType.TextProposal, proposer:myAddress, initial_deposit:stringToCoins("100000000uscrt"), content:{ title:"Hi", description:"Let's vote on this", }, }, { gasLimit:50_000, }, );

constproposalId=Number( tx.arrayLog.find( (log)=>log.type==="submit_proposal"&&log.key==="proposal_id", ).value, );

```

secretjs.tx.gov.submitProposal.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.gov.vote()

MsgVote defines a message to cast a vote.

Input:MsgVoteParams

```

Copy consttx=awaitsecretjs.tx.gov.vote( { voter:myAddress, proposal_id:someProposalId, option:VoteOption.VOTE_OPTION_YES, }, { gasLimit:50_000, }, );

```

secretjs.tx.gov.vote.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.gov.voteWeighted()

MsgVoteWeighted defines a message to cast a vote, with an option to split the vote.

Input:MsgVoteWeightedParams

```

Copy // vote yes with 70% of my power consttx=awaitsecretjs.tx.gov.voteWeighted( { voter:myAddress, proposal_id:someProposalId, options:[ // weights must sum to 1.0 { weight:0.7,option:VoteOption.VOTE_OPTION_YES}, { weight:0.3,option:VoteOption.VOTE_OPTION_ABSTAIN}, ], }, { gasLimit:50_000, }, );

```

secretjs.tx.gov.voteWeighted.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.ibc.transfer()

MsgTransfer defines a msg to transfer fungible tokens (i.e Coins) between ICS20 enabled chains. See ICS Spec here:https://github.com/cosmos/ics/tree/master/spec/ics-020-fungible-token-transfer#data-structures

Input:MsgTransferParams

secretjs.tx.ibc.transfer.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.slashing.unjail()

MsgUnjail defines a message to release a validator from jail.

Input:MsgUnjailParams

```

Copy consttx=awaitsecretjs.tx.slashing.unjail( { validator_addr:mValidatorsAddress, }, { gasLimit:50_000, }, );

```

secretjs.tx.slashing.unjail.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.staking.beginRedelegate()

MsgBeginRedelegate defines an SDK message for performing a redelegation of coins from a delegator and source validator to a destination validator.

Input:MsgBeginRedelegateParams

```

Copy consttx=awaitsecretjs.tx.staking.beginRedelegate( { delegator_address:myAddress, validator_src_address:someValidator, validator_dst_address:someOtherValidator, amount:stringToCoin("1uscrt"), }, { gasLimit:50_000, }, );

```

secretjs.tx.staking.beginRedelegate.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.staking.createValidator()

MsgCreateValidator defines an SDK message for creating a new validator.

Input:MsgCreateValidatorParams

```
Copy consttx=awaitsecretjs.tx.staking.createValidator( { delegator_address:myAddress, commission:{
max_change_rate:0.01,// can change +-1% every 24h max_rate:0.1,// 10% rate:0.05,// 5% }, description:{ moniker:"My
validator's display name", identity:"ID on keybase.io, to have a logo on explorer and stuff", website:"example.com",
security_contact:"hi@example.com", details:"We are good", }, pubkey:toBase64(newUint8Array(32).fill(1)),// validator's
pubkey, to sign on validated blocks min_self_delegation:"1",// uscrt initial_delegation:stringToCoin("1uscrt"), }, {
gasLimit:100_000, }, );
```

secretjs.tx.staking.createValidator.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.staking.delegate()

MsgDelegate defines an SDK message for performing a delegation of coins from a delegator to a validator.

Input:MsgDelegateParams

```
Copy consttx=awaitsecretjs.tx.staking.delegate( { delegator_address:myAddress, validator_address:someValidatorAddress,
amount:stringToCoin("1uscrt"), }, { gasLimit:50_000, }, );
```

secretjs.tx.staking.delegate.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.staking.editValidator()

MsgEditValidator defines an SDK message for editing an existing validator.

Input:MsgEditValidatorParams

```
Copy consttx=awaitsecretjs.tx.staking.editValidator( { validator_address:myValidatorAddress, description:{ // To edit even
one item in "description you have to re-input everything moniker:"papaya", identity:"banana", website:"watermelon.com",
security_contact:"sec@watermelon.com", details:"We are the banana papaya validator yay!", }, min_self_delegation:"2",
commission_rate:0.04,// 4%, commission cannot be changed more than once in 24h }, { gasLimit:5_000_000, }, );
```

secretjs.tx.staking.editValidator.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .

secretjs.tx.staking.undelegate()

MsgUndelegate defines an SDK message for performing an undelegation from a delegate and a validator

Input:MsgUndelegateParams

```
Copy consttx=awaitsecretjs.tx.staking.undelegate( { delegator_address:myAddress,
validator_address:someValidatorAddress, amount:stringToCoin("1uscrt"), }, { gasLimit:50_000, }, );
```

secretjs.tx.staking.undelegate.simulate()

Simulates execution without sending a transactions. Input is exactly like the parent function. For more info seesecretjs.tx.simulate() .