Auto-wrapping of SNIP-20 tokens with IBC Hooks

A step-by-step guide on how to auto-wrap SNIP-20 tokens with IBC hooks.

Tutorial: Auto-wrapping of SNIP-20 Tokens

In this tutorial, you will learn how to usedIBC hooks to auto-wrapSNIP-20 tokens between two LocalSecret IBC chains. Simply put, you will learn how to IBC transfer tokens from one blockchain to another blockchain, and in doing so, the token transfer will execute a smart contract call that turns the tokens into privacy-preserving tokens, all with a single token transfer!

Overview

What You Will Learn: This tutorial is structured to cover the following key steps:

1. Setting up a Hermes relayer between two LocalSecret chains, specifically Chainsecretdev-1
2. and Chainsecretdev-2
3. .
4. Executing an IBC token transfer between the two aforementioned chains.
5. Uploading and initializing a SNIP-20 contract on the secretdev-1 chain.
6. Uploading and initializing a Wasm Hooks wrapper contract on the secretdev-1 chain.
7. Transferring tokens from secretdev-1 to secretdev-2, and in the process, utilizing IBC hooks to automatically wrap the SNIP-20 tokens.
8.

Let's dive in! ♀

Set up Hermes Relayer

To follow along with this tutorial, setup a Hermes relayer between two LocalSecret chains using theSecret IBC setup documentation . Once you have established anIBC transfer channel , you are ready to proceed to the next step.

Execute IBC Token Transfer

In order to auto-wrap SNIP-20 tokens over IBC, you must first instantiate a SNIP-20 smart contract on LocalSecret with theIBC denom of the token.

Want to learn more about IBC denoms? Read theCosmos documentation here. You can find the IBC denom of your token by executing an IBC token transfer. Assuming you have funded the LocalSecret wallet'a' for chain secretdev-1, run the following:

```

Copy

# be on the source network (secretdev-1)

secretcliconfignodehttp://localhost:26657

# check the initial balance of a

secretcliqbankbalancessecret1ap26qrlp8mcq2pg6r47w43l0y8zkqm8a450s03

# transfer to the destination network

secretclitxibc-transfertransfertransferchannel-0secret1ap26qrlp8mcq2pg6r47w43l0y8zkqm8a450s031uscrt--froma

```

If you have never funded a Hermes wallet or LocalSecret wallet beforelearn how to do so here .

If you would like to do further reading, refer to theLocalSecret wallet docs and theHermes wallet docs . Query that the transaction was successful:

```

Copy secretcliquerytx

```

Now switch to the destination network (secretdev-2) and query the bank balance of wallet'a'

to confirm that the transaction was successful:

```

Copy

# switch to the destination network (secretdev-2)

secretcli config node http://localhost:36657

# check that you have an ibc-denom

secretcli q bank balances secret1ap26qrlp8mcq2pg6r47w43l0y8zkqm8a450s03

```

Congrats You should now see the ibc-denom returned in the transaction query:

```

Copy {"balances":[{"denom":"ibc/834829648E6B51B21713C76E0C1836727DCE221CE3DC8B3AA7BB11F55428887A","amount":"1"},{"denom":"uscrt","amount":"9999999999998611996"}],"pagination":{"next_key":null,"total":"0"}}

```

SNIP-20 Contract

Now, let's upload the SNIP-20 contract and instantiate it with the correct IBC denom.cd into/contracts/snip20-reference-impl and then compile the contract:

```

Copy dockerrun--rm-v"(pwd)":/contract\ --mounttype=volume,source="(basename"(pwd)")_cache",target=/code/target\ --mounttype=volume,source=registry_cache,target=/usr/local/cargo/registry\ enigmampc/secret-contract-optimizer

```

Now upload it (but first make sure you are on chain secretdev-1):

```

Copy secretcliconfignodehttp://localhost:26657
```

```

```

Copy secretclitxcomputestorecontract.wasm.gz--gas5000000--froma--chain-idsecretdev-1

```

Query that the upload was successful:

```

Copy secretcliquerycomputelist-code

```

If the upload was successful it should return:

```

Copy [{ "code_id":1, "creator":"secret1ap26qrlp8mcq2pg6r47w43l0y8zkqm8a450s03", "code_hash":"c74bc4b0406507257ed033caa922272023ab013b0c74330efc16569528fa34fe" }]

```

Lastly, instantiate the contract with your associated IBC denom:

```

Copy random_bytes=(opensslrand-base6432) secretclitxcomputeinstantiate1'{ "name": "Secret Secret", "symbol": "sSCRT", "decimals": 6, "prng_seed": ""random_bytes"", "admin": "secret1ap26qrlp8mcq2pg6r47w43l0y8zkqm8a450s03", "initial_balances": [{ "address": "secret1ap26qrlp8mcq2pg6r47w43l0y8zkqm8a450s03", "amount": "1000000000" }], "supported_denoms": ["ibc/834829648E6B51B21713C76E0C1836727DCE221CE3DC8B3AA7BB11F55428887A"] }'--froma--labelsnip20-y

```

Query successful instantiation:

```

Copy secretcliquerycomputelist-contract-by-code1

```

If the instantiation was successful, it will return:

```

Copy [{ "contract_address":"secret1mfk7n6mc2cg6lznujmeckdh4x0a5ezf6hx6y8q", "code_id":1, "creator":"secret1ap26qrlp8mcq2pg6r47w43l0y8zkqm8a450s03", "label":"snip20" }]

```

Wasm Hooks Contract

Now let's upload and instantiate the wasm hooks contract First,cd into ./contracts/ibc-hooks-contract

Upload:

```

Copy secretclitxcomputestorecontract.wasm.gz--gas5000000--froma--chain-idsecretdev-1

```

Query successful upload:

```

Copy secretcliquerycomputelist-code

```

Upon successful upload it will return:

```

Copy { "code_id":2, "creator":"secret1ap26qrlp8mcq2pg6r47w43l0y8zkqm8a450s03", "code_hash":"951cbc1b87b3d360f9a18aaf3fe152a2777bd0d5b595c1509f0980ef45441a36" }

```

Instantiate:

```

Copy secretclitxcomputeinstantiate2'{}'--froma--labelwrap-ibc-y

```

Query successful instantiation:

```

Copy secretcliquerycomputelist-contract-by-code2

```

If the instantiation was successful it will return:

```

Copy [{ "contract_address":"secret1gyruqan6yxf0q423t8z5zce3x7np35uw8s8wqc", "code_id":2, "creator":"secret1ap26qrlp8mcq2pg6r47w43l0y8zkqm8a450s03", "label":"wrap-ibc" }]

```

Auto-wrap Tokens

Now all that's left is to make an IBC token transfer and experience the magic of IBC hooks with the token wrapping contract.

First, initialize the variables in your terminal:

```

Copy HUB_CHAIN_ID="secretdev-1"

sSCRT="secret1mfk7n6mc2cg6lznujmeckdh4x0a5ezf6hx6y8q"

WRAP_DEPOSIT_CONTRACT_ADDRESS="secret1gyruqan6yxf0q423t8z5zce3x7np35uw8s8wqc"

myScrtAddress="secret1ap26qrlp8mcq2pg6r47w43l0y8zkqm8a450s03"

memo=(echo-n'{"wasm":{"contract":"'WRAP_DEPOSIT_CONTRACT_ADDRESS'","msg":{"wrap_deposit":{"snip20_address":"'sSCRT'","recipient_address":"'myScrtAddress'"}}}}'|base64)
```
```

```

Now execute the token transfer!

```

Copy secretclitxibc-transfertransfertransferchannel-0"sSCRT"1uscrt--memo"memo"--froma

```

Query that it was successful:

```

Copy secretcliquerytx

```

Here is an excerpt from the transaction query which includes the IBC memo:

```

Copy [{"key":"recipient","value":"secret1a53udazy8ayufvy0s434pfwjcedzqv345fkvkj"},{"key":"sender","value":"secret1ap26qrlp8mcq2pg6r47w43l0y8zkqm8a450s03"},
{"key":"amount","value":"1uscrt"}]}]}],"info":"","gas_wanted":"200000","gas_used":"19809","tx":{"@type":"/cosmos.tx.v1beta1.Tx","body":{"messages":
[{"@type":"/ibc.applications.transfer.v1.MsgTransfer","source_port":"transfer","source_channel":"channel-0","token":
{"denom":"uscrt","amount":"1"},"sender":"secret1ap26qrlp8mcq2pg6r47w43l0y8zkqm8a450s03","receiver":"secret1mfk7n6mc2cg6lznujmeckdh4x0a5ezf6hx6y8q","timeout_height":
{"revision_number":"2","revision_height":"1483"},"timeout_timestamp":"1694102077056751000","memo":"eyJ3YXNtIjp7ImNvbnRyYWN0Ijoic2VjcmV0MWd5cnVxYW42eXhmMHE0MjN0OHo1emNlM3g3t

```

Notice that the sender is wallet address'a'

and the receiver is theSNIP-20 contract address.

Congrats! You've just successfully used IBC hooks to auto-wrap tokens on Secret Network

Last updated6 months ago On this page *Tutorial: Auto-wrapping of SNIP-20 Tokens * Overview * Set up Hermes Relayer * Execute IBC Token Transfer * SNIP-20 Contract * Wasm Hooks Contract * Auto-wrap Tokens

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