Cross-deploy Vanilla CW and Secret Contracts

A description of building the same code for both vanilla CosmWasm and the Secret version.

Crosschain Contract Demo

The crosschain contract that we are examining can be cloned fromthis repo . This repo is a demo of writing a simple voting contract that is intended to be deployed on multiple chains with different types of CosmWasm in asingle codebase . This demo supports two types of CosmWasm:vanilla (i.e. original, like Terra or Juno supported) andsecret (i.e. SecretNetwork supported).

The contract code uses conditional compilation and feature flags in Rust to support different configurations within a single codebase. If you navigate tocontract.rs , you will find two sets of imports for different features, either"secret" or"vanilla" . The imports are conditionally compiled using#[cfg(feature = "…")] attributes. The"secret" feature uses thesecret_std crate, whereas the"vanilla" feature uses thecosmwasm_std crate.

For more information about the feature differences of Standard CosmWasm vs the secret version please visitthis page.
TLDR : No raw queries, iterators and upgradeability

Contract.rs

Contract.rs defines several functions:

1. instantiate
2. : This function initializes the contract and sets the initial vote count for both options to 0.In this instance, it is the same for both Secret and vanilla CosmWasm.
3. execute
4. : This function processes theExecuteMsg
5. , which supports voting. It calls thevote_impl
6. function to perform the vote.
7. vote_impl
8. : This function is implemented twice, once for each feature ("secret"
9. and"vanilla"
10.). It checks whether the voter has already voted, and if not, it updates the tally for the chosen option.
11. query
12. : This function processesQueryMsg
13. , which supports two queries: getting the vote tallies (Tally
14.) and getting a list of voters (Voters
15.). For theTally
16. query, it returns the vote count for both options. For theVoters
17. query, it calls thevoters_query_impl
18. function.
19. voters_query_impl
20. : This function is also implemented twice for each feature. It retrieves the voters list based on the provided page number.
21.

Let's examine the differences invote_impl based on which type of CosmWasm we are using. The overall structure and logic of the function are the same, but there are differences in the specific methods called on theVOTERS ,OPTION_1 , andOPTION_2 objects.These differences arise from the different crates used for the "secret" and "vanilla" features.

fn vote_impl: Secret

In the "secret" version ofvote_impl :

1. VOTERS.contains(deps.storage, &info.sender)
2. is used to check if the voter already exists in the storage.
3. VOTERS.insert(deps.storage, &info.sender, &1)
4. is used to insert the voter into the storage.
5.

```

Copy

# [cfg(feature="secret")]

fnvote_impl(deps:DepsMut, info:MessageInfo, option:u64)->StdResult { ifVOTERS.contains(deps.storage,&info.sender) {
```

returnErr(StdError::generic_err("already voted")); }

VOTERS.insert(deps.storage,&info.sender,&1/ *arbitrary value* /)?;

// Update tally matchoption { 1=>OPTION_1.update(deps.storage,|tally|Ok(tally+1))?,
2=>OPTION_2.update(deps.storage,|tally|Ok(tally+1))?, _=>returnErr(StdError::generic_err("unsupported option")), };

Ok(Response::default()) }

```

fn vote_impl: Vanilla CosmWasm

In the "vanilla" version ofvote_impl :

1. VOTERS.has(deps.storage, info.sender.clone())
2. is used to check if the voter already exists in the storage.
3. VOTERS.save(deps.storage, info.sender, &1)
4. is used to save the voter into the storage.
5.

```

Copy

[cfg(feature="vanilla")]

fnvote_impl(deps:DepsMut, info:MessageInfo, option:u64)->StdResult { ifVOTERS.has(deps.storage, info.sender.clone()) {
returnErr(StdError::generic_err("already voted")); }

VOTERS.save(deps.storage, info.sender,&1/ *arbitrary value* /)?;

// Update tally matchoption { 1=>OPTION_1.update(deps.storage,|tally|Ok:{tally+1))?,
2=>OPTION_2.update(deps.storage,|tally|Ok::(tally+1))?, _=>returnErr(StdError::generic_err("unsupported option")), };

Ok(Response::default()) }

```

The rest of the function, including the match statement for updating the vote tally, is the same between the two versions. Now let's examine the difference in querying functions between the two versions.

fn voters_query_impl: Secret

Vanilla CosmWasm Iterators are not supported on Secret Network because users cannot iterate over data stored by different usersas there is no access to their dedicated encryption key. However, iteration is still possible on Secret Network through the use of thesecret_toolkit for storage in place ofcosmwasm_std .

In the "secret" version ofvoters_query_impl :

- TheVOTERS.paging_keys(deps.storage, page, PAGE_SIZE as u32)?
- method is used to retrieve a list of voters corresponding to the requested page. This method is specific to thesecret_toolkit
- storage API and directly handles pagination.
-

```

Copy

[cfg(feature="secret")]

fnvoters_query_impl(deps:Deps, page:u32)->StdResult { letvoters=VOTERS.paging_keys(deps.storage, page,
PAGE_SIZEasu32)?; Ok(QueryRes::Voters{ voters }) }

```

fn voters_query_impl: Vanilla
```

In the "vanilla" version ofvoters_query_impl :

- TheVOTERS.keys(deps.storage, None, None, cosmwasm_std::Order::Ascending)
- method is used to retrieve an iterator over all the keys (voters) in the storage. Pagination is implemented manually in the function.
-

```

Copy

# [cfg(feature="vanilla")]

fnvoters_query_impl(deps:Deps, page:u32)->StdResult { usecosmwasm_std::Addr;

letvoters_iter=VOTERS.keys(deps.storage,None,None, cosmwasm_std::Order::Ascending);//.paging_keys(deps.storage, page, 20)?; letvoters:Vec=voters_iter .skip((pageasusize)*PAGE_SIZE) .take(PAGE_SIZE) .filter(|v|v.is_ok()) .map(|v|v.unwrap()) .collect(); Ok(QueryRes::Voters{ voters:voters }) }

```

The main difference between the two implementations is that the Secret version relies on asecret_toolkit pagination method (paging_keys), whereas the Vanilla version manually implements pagination using iterator methods.

State.rs: Secret vs CosmWasm

The contract usesstate_secret orstate_vanilla modules for managing the state, depending on the selected feature. The state management includes saving the vote counts for each option and managing the list of voters. Let's examine the differences betweenstate_secret.rs andstate_vanilla.rs .

state_secret.rs

In the Secret version:

1. Thesecret_std::Addr
2. is used as the address type.
3. Thesecret_toolkit::storage::Item
4. andsecret_toolkit::storage::Keymap
5. types are used for storage management.
6. The storage objects are initialized withKeymap::new
7. andItem::new
8. methods, passing the byte representation of the corresponding prefixes.
9.

```

Copy

# ![cfg(feature="secret")]

usecrate::state::{OPTION_1_PREFIX,VOTE_PREFIX}; usesecret_std::Addr; usesecret_toolkit::storage::{Item,Keymap};

pubstaticVOTERS:Keymap=Keymap::new(VOTE_PREFIX.as_bytes());
pubstaticOPTION_1:Item=Item::new(OPTION_1_PREFIX.as_bytes());
pubstaticOPTION_2:Item=Item::new(OPTION_1_PREFIX.as_bytes());rust

```

state_vanilla.rs

In the Vanilla version:

1. Thecosmwasm_std::Addr
2. is used as the address type.
3. Thecw_storage_plus::Item
4. andcw_storage_plus::Map
5. types are used for storage management.
6. The storage objects are initialized with theMap::new

7. andItem::new
8. methods, passing the corresponding prefixes directly.
9.

```

Copy

# ![cfg(feature="vanilla")]

usecrate::state::{OPTION_1_PREFIX,VOTE_PREFIX}; usecosmwasm_std::Addr; usecw_storage_plus::{Item,Map};

pubstaticVOTERS:Map=Map::new(VOTE_PREFIX); pubstaticOPTION_1:Item=Item::new(OPTION_1_PREFIX); pubstaticOPTION_2:Item=Item::new(OPTION_1_PREFIX);

```

Thus, the Secret version relies on thesecret_std andsecret_toolkit crates, while the Vanilla version uses thecosmwasm_std andcw_storage_plus crates. However, the overall purpose of the state management objects (VOTERS, OPTION_1, and OPTION_2) remains the same in both versions.

How to cross-deploy on different chains

```

Copy //Building for Secret make secret

//Building for Juno make vanilla

```

Last updated7 months ago On this page *Crosschain Contract Demo * Contract.rs * State.rs: Secret vs CosmWasm * How to cross-deploy on different chains

Was this helpful? Edit on GitHub Export as PDF