Secret Network v1.4 (CosmWasm 1.0)

CosmWasm 1.0 Breaking Changes

Address API Changes

- HumanAddr
- has been deprecated in favour of simplyString
- . It never added any significant safety bonus overString
- and was just a marker type. The new typeAddr
- was created to hold validated addresses. Those can be created viaAddr::unchecked
- ,Api::addr_validate
- ,Api::addr_humanize
- and JSON deserialization. In order to maintain type safety, deserialization intoAddr
- must only be done from trusted sources like a contract's state or a query response. User inputs must be deserialized intoString
- .
- deps.api.human_address(&CanonicalAddr)
- =>deps.api.addr_humanize(&CanonicalAddr)
- deps.api.canonical_address(&HumanAddr)
- =>deps.api.addr_canonicalize(&str)
-

Extern Method Interface Changes

Use the new entry point system. From lib.rs remove

```

Copy

# [cfg(target_arch="wasm32")]

cosmwasm_std::create_entry_points!(contract);

// or

# [cfg(target_arch="wasm32")]

cosmwasm_std::create_entry_points_with_migration!(contract);

```

Then add the macro attribute#[cfg_attr(not(feature = "library"), entry_point)] to your contract.rs as follows:

- init
-
  - Env
-
  - split intoEnv
-
  - andMessageInfo
-
  - InitResponse
-
  - andInitResult
-
  - deprecated, please useResponse
-
  - function name changed frominit
-
  - toinstantiate
- *
-

```

Copy pubfninit( deps:&mutExtern, env:Env, msg:InitMsg, )->StdResult {

// into

# [cfg_attr(not(feature="library"), entry_point)]

pubfninstantiate( deps:DepsMut, env:Env, info:MessageInfo, msg:InstantiateMsg, )->StdResult {

```

- handle
-

-
    - Env
-
    - split intoEnv
-
    - andMessageInfo
-
    - HandleResponse
-
    - andHandleResult
-
    - deprecated, please useResponse
-
    - function name changed fromhandle
-
    - toexecute
- *
-

```

Copy pubfnhandle( deps:&mutExtern, env:Env, msg:HandleMsg, )->HandleResult{

// into

# [cfg_attr(not(feature="library"), entry_point)]

pubfnexecute(deps:DepsMut, env:Env, info:MessageInfo, msg:ExecuteMsg)->StdResult {

```

- query
-
    - new argumentEnv
-
    - added
- *
-

```

Copy pubfnquery( deps:&Extern, msg:QueryMsg, )->StdResult {

// into

# [cfg_attr(not(feature="library"), entry_point)]

pubfnquery(deps:Deps, _env:Env, msg:QueryMsg)->StdResult {

```

- migrate
-
    - Env
-
    - split intoEnv
-
    - andMessageInfo
-
    - MigrateResponse
-
    - andMigrateResult
-
    - deprecated, please useResponse
- *
-

```

Copy pubfnmigrate( deps:&mutExtern, env:Env, msg:MigrateMsg, )->MigrateResult{

// into

# [cfg_attr(not(feature="library"), entry_point)]

pubfnmigrate(_deps:DepsMut, _env:Env, _msg:MigrateMsg)->StdResult {

```

Response no longer be built using a structure literal

Response can no longer be built using a struct literal. Please useResponse::new as well as relevantbuilder-style setters to set the data.

This is a step toward better API stability.

```

Copy letsend=BankMsg::Send{ to_address, amount };

-Ok(Response{ -messages:vec![SubMsg::new(send)], -attributes:vec![attr("action","burn"),attr("payout", msg.payout)], -events:vec![], -data:Some(data_msg.into()), -}) +Ok(Response::new() +.add_message(send) +.add_attribute("action","burn") +.add_attribute("payout", msg.payout) +.set_data(data_msg))

```

Sub-messages & Reply

The sub-messages feature can be used to get the response data or events from the executed contract. For example, if a contract wants to get the address of the child contract which is instantiated from the contract. The contract can sendMsgInstantiate as sub-messages withReplyOn::Success option likehttps://github.com/terraswap/terraswap/blob/7cf47f5e811fe0c4643a7cd09500702c1e7f3a6b/contracts/terraswap_factory/src/contract.rs#L128-L142 .

Then the reply is only executed when the instantiate is successful with the instantiate response data.https://github.com/terraswap/terraswap/blob/7cf47f5e811fe0c4643a7cd09500702c1e7f3a6b/contracts/terraswap_factory/src/contract.rs#L148-L170 .

Storage Migration

Rename the typeExtern toDeps , and radically simplify theinit/handle/migrate/query entrypoints. Rather than&mut Extern, useDepsMut . And instead of&Extern , useDeps . If you ever pass eg.foo(api: A) around, you must now use dynamic trait pointers:foo(api: &dyn Api) . Here is the quick search-replace guide on how to fix contract.rs:

In production (non-test) code:

```

Copy =>`` &mutExtern=>DepsMut &Extern=>Deps &mutdeps.storage=>deps.storagewherepassing into state.rs helpers &deps.storage=>deps.storagewherepassing into state.rs helpers Onthe top, removeusecosmwasm_std::{Api,Extern,Querier,Storage}.Addusecosmwasm_std::{Deps,DepsMut}.

```

In test code only:

```

Copy &mutdeps,=>deps.as_mut(), &deps,=>deps.as_ref(), Youmay have to addusecosmwasm_std::{Storage}ifthe compile complains about thetrait

```

If you use cosmwasm-storage, in state.rs:

```

Copy =><S:ReadonlyStorage>=> < &mutS=>&mutdynStorage &S=>&dynStorage Ifyou have any references toReadonlyStorageleft after the above, please replace them withStorage

```

Advanced Storage

We can still usesingleton andbucket . But if you want more advanced storage access, you can usecw-storage-plus with following migration steps.

- cowmasm_storage::Singleton
- ->cw_stroage_plus::Item
-
    - Removeread_*
-
    - andstore_*
-
    - functions
-
    - DefineItem
-
    - as following (must prepend the length of key)
-
    -

-
  ```
- Copy
- pub const CONFIG: Item = Item::new("\u{0}\u{6}config");
- // store
- CONFIG.save(deps.storage, &config)?;
- // read
  ```

- let mut config: Config = CONFIG.load(deps.storage)?;

-
  ```
- cosmwasm_storage::Bucket
- ->cw_storage_plus::Map
-
  - Removeread_*
-
  - andstore_*
-
  - functions
-
  - DefineMap
-
  - as following
-
  -
- ```
- Copy
- pub const PAIRS: Map = Map::new("pair_info");
- // store
- PAIRS.save(deps.storage, &addr, &pair_info)?;
- // read
- let pair_info: PairInfoRaw = PAIRS.load(deps.storage, &addr)?;
- ```
-

Raw Querier migration

The core now just returns raw bytes without json encoding, so we can receive the query response as what the data was stored.

```

Copy -letres:Binary=deps.querier.query(&QueryRequest::Wasm(WasmQuery::Raw{ ... -letpair_info:PairInfoRaw=from_binary(&res)?;

// into

+letpair_info:PairInfoRaw=deps.querier.query(&QueryRequest::Wasm(WasmQuery::Raw{ ...

```

Also,mock_querier has to remove oneto_binary from its raw query response.

Last updated7 months ago On this page *CosmWasm 1.0 Breaking Changes * Address API Changes * Extern Method Interface Changes * Response no longer be built using a structure literal* Sub-messages & Reply * Storage Migration * Advanced Storage * Raw Querier migration

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