TFL Smart Contract Project: Message Hydration

Goal

In Warp, one of the most important functions is serializing and deserializing data between strings and objects. In some cases, we hydrate messages with real-time parameters that are unknown during job creation. Typically, within contracts, messages are formatted using base64. In some cases, there are multiple levels of nested base64 encodings within messages. In particular, this can be seen in a Cw20ExecuteMsg::Send message. In this case both the top level WasmMsg msg field, well as the msg field inside of the CW20 send are encoded using base64.
The goal of this assignment is to create a generic algorithm in Rust and CosmWasm that looks for variables within stringified messages, then properly encodes and converts these messages into a functional CosmosMsg.
The variables are inputted using an array and are noted as $warp.var.VARIABLE_NAME. Some may be contained within the nested base64 encoded strings. An example of a message that must be properly decoded, hydrated, and encoded again is found in the Appendix.

Expectations

- 2-4 hours spent on the smart contract
- Clear README.md describing how to run and test the contract
- Clear comments within the code describing the algorithm

Requirements

- The CosmWasm version must be 1.0
- The code must be generic and work with any number of nested base64 binary messages
- The execute message must take the following parameters:
    - msg: String
    - vars: String
- The execute message must send off a fully functional CosmosMsg in the Response
- The serde_json_wasm library must be used to convert between strings and Rust objects
- The code for the software is to be hosted on GitHub

Resources

- Terra Developer Docs: https://docs.terra.money/
- CosmWasm: https://docs.cosmwasm.com/docs/
- serde_json_wasm: https://github.com/CosmWasm/serde-json-wasm
- CosmosMsg: https://docs.rs/cosmwasm-std/latest/cosmwasm_std/enum.CosmosMsg.html

Appendix

INPUT STRING:


```
{
  "wasm": {
    "execute": {
      "contract_addr": "$warp.var.variable1",
      "msg":
"eyJzZW5kIjp7ImNvbnRyYWN0IjoidGVycmE1NDMyMSIsImFtb3VudCI6IjEyMzQ1IiwibXNnIjoiZXlKbGVHV
mpkWFJsWDNOM1lYQmZiM0JsY21GMGFXOXVjeUk2ZXlKdmNHVnlZWFJwYjI1eklqcGJleUpoYzNSeWIxOXpkMkZ
3SWpwN0ltOW1abVZ5WDJGemMyVjBYMmx1Wm04aU9uc2lkRzlyWlc0aU9uc2lZMjl1ZEhKaFkzUmZZV1JrY2lJN
klpUjNZWEp3TG5aaGNpNTJZWEpwWVdKc1pURWlmWDBzSW1GemExOWhjM05sZEY5cGJtWnZJanA3SW01aGRHbDJ
aVjkwYjJ0bGJpSTZleUprWlc1dmJTSTZJaVIzWVhKd0xuWmhjaTUyWVhKcFlXSnNaVElpZlgxOWZWMHNJbTFwY
m1sdGRXMWZjbVZqWldsMlpTSTZJaVIzWVhKd0xuWmhjaTUyWVhKcFlXSnNaVE1pTENKMGJ5STZJaVIzWVhKd0x
uWmhjaTUyWVhKcFlXSnNaVFFpTENKdFlYaGZjM0J5WldGa0lqb2lKSGRoY25BdWRtRnlMblpoY21saFlteGxOU
0o5ZlE9PSJ9fQ==",
      "funds": []
    }
  }
}
```


INPUT VARIABLES


```
[
  "$warp.var.variable1": "terra12345",
  "$warp.var.variable2": “uterra",
  "$warp.var.variable3": “54321",
  "$warp.var.variable4": "terra11111",
  "$warp.var.variable5": "0.05",
]
```


OUTPUT STRING


```
{
  "wasm": {
    "execute": {
      "contract_addr": "terra12345",
      "msg":
"eyJzZW5kIjp7ImNvbnRyYWN0IjoidGVycmE1NDMyMSIsImFtb3VudCI6IjEyMzQ1IiwibXNnIjoiZXlKbGVHV
mpkWFJsWDNOM1lYQmZiM0JsY21GMGFXOXVjeUk2ZXlKdmNHVnlZWFJwYjI1eklqcGJleUpoYzNSeWIxOXpkMkZ
3SWpwN0ltOW1abVZ5WDJGemMyVjBYMmx1Wm04aU9uc2lkRzlyWlc0aU9uc2lZMjl1ZEhKaFkzUmZZV1JrY2lJN
kluUmxjbkpoTVRJek5EVWlmWDBzSW1GemExOWhjM05sZEY5cGJtWnZJanA3SW01aGRHbDJaVjkwYjJ0bGJpSTZ
leUprWlc1dmJTSTZJblYwWlhKeVlTSjlmWDE5WFN3aWJXbHVhVzExYlY5eVpXTmxhWFpsSWpvaU5UUXpNakVpT
ENKMGJ5STZJblJsY25KaE1URXhNVEVpTENKdFlYaGZjM0J5WldGa0lqb2lNQzR3TlNKOWZRPT0ifX0=",
      "funds": []
    }
  }
}
```