Oracle WebAssembly (Owasm)

Oracle WebAssembly, or Owasm for short, is Band Protocol's Domain Specific Language (DSL) on top of the Rust programing language for writing oracle scripts oracle scripts to be used in the BandChain ecosystem. The Owasm library consists of two primary modules:

owasm/oei and owasm/ext.

 Owasm/OEI - The OEI modules defines a set of functions that are part of the Owasm Oracle Environment Interface. These

By using this website, you agree to our Cookie Policy. X

script during its execution. The complete list

of functions and implementation can be found here \nearrow .

Owasm/Ext - The Owasm extension module provides a convenient way to write oracle scripts that connect to various public APIs.
 Examples of these are functions to calculate the mean, median, and majority values from the validator's reported results, which can be used during the aggregation phase of an oracle script. The full list of functions and its implementation can be found here

Usage

To illustrate an example usage of the Owasm library, we will be using the example below. The code is based off an oracle script for retrieving the price of a stock.

```
use obi::{OBIDeserialize, OBISchema, OBISerialize}
use owasm::{execute entry point, ext, oei, prepare
#[derive(OBIDeserialize, OBISchema)]
struct Input {
    symbol: String,
   multiplier: u64,
#[derive(OBISerialize, OBISchema)]
struct Output {
    px: u64,
#[no_mangle]
fn prepare impl(input: Input) {
    oei::ask external data(14, 1, &input.symbol.as
#[no mangle]
fn execute impl(input: Input) -> Output {
    let avg: f64 = ext::load average(1);
    Output { px: (avg * input.multiplier as f64) as
prepare entry point!(prepare impl);
execute entry point!(execute impl);
```

The script starts off by defining the input and output structs. In this case, the input comprises of the stock ticker/symbol (string) and the

By using this website, you agree to our Cookie Policy. X
by (u64). On the other hand, the output is

simply the price of the stock multiplied by the multiplier, returned as a [u64] value.

Once the structs and types of both input and output have been determined, we move on to defining the preparation and execution phases of the oracle script, defined by prepare_impl and execute_impl, respectively.

In order to call these functions, we need to pass appriopriate input values and make the function calls. To do so, oracle script writer can use our macros defined in macros.rs , also shown below. The aim of these macros is to reduce the load of the script writer by handling the work of retrieving the calldata, deserializing it, and using it to construct the appropriate input struct for them.

```
#[macro export]
macro rules! prepare entry point {
    ($name:ident) => {
        #[no_mangle]
        pub fn prepare() {
            $name(OBIDeserialize::try from slice(&c
    };
#[macro export]
macro_rules! execute_entry_point {
    ($name:ident) => {
        #[no mangle]
        pub fn execute() {
            oei::save_return_data(
                &$name(OBIDeserialize::try from sl:
                    .try to vec()
                    .unwrap(),
            );
    };
```

The last two lines of the oracle script above shows the macros in action.

Preparation phase

The prepare_impl function takes the previously-

By using this website, you agree to our Cookie Policy.





DOCUMENTATION

Introduction

Whitepaper

Introduction

Terminology

System Overview

Token Economics

Protocol Messages

Decentralized Validator Sampling

Oracle WebAssembly (Owasm)

Lite Client Protocol

Cosmos IBC Integration

On-chain Payment Protocol

Example Use Cases

Technical Specifications

BAND STANDARD DATASET

♦ Introduction

Supported Blockchains

function then has only one main task; calling the ask_external_data > function in oei module.

In case extra information from relating oracle request is needed for implementing some logic before calling ask_external_data, here are functions and their details from oei module which can be used in prepare_impl.

- get_ask_count() returns i64
 - Returns the number of validators to asked to report for this oracle request.
- [get_min_count() | returns | i64
 - Returns the minimum number of data reports as specified by the oracle request.
- get_calldata() returns Vec<u8>
 - Returns the raw calldata as specified when the oracle request is submitted.
 - Note: This function is already called in macros when preparing the input.

By using this website, you agree to our Cookie Policy.

Search

ON THIS PAGE

Usage

Preparation phase

Execution phase

- ask_external_data(eid: i64, did: i64, calldata: &[u8])
 - Takes in external id, data source id, and data in bytes.
 - Issues a new raw request to the host environment using the specified data source ID and calldata, and assigns it to the given external ID.

Execution phase

The execute_impl function takes in the input type as an argument as well, but also returns the output struct type. It then starts by computing the final value of the request through calling load_average > function from the ext module and proceed to use the computed average to construct and return the appropriate output struct.

Below is the list of functions from oei that can be called in execute_impl.

• get_ask_count() returns i64

By using this website, you agree to our Cookie Policy.

asked to report for this oracle request.

- get_min_count() returns i64
 - Returns the minimum number of data reports as specified by the oracle request.
- get_ans_count() returns i64
 - Returns the number of validators that report data to this oracle request. Must only be called during execution phase.
- get_calldata() returns Vec<u8>
 - Returns the raw calldata as specified when the oracle request is submitted.
 - Note: This function is already called in macros when preparing the input.
- save_return_data(data: &[u8])
 - Saves the given data as the result of the oracle execution. Must only be called during execution phase and must be called exactly once.

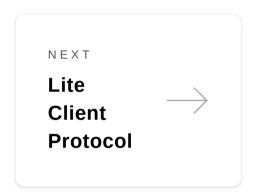
 Returns the data reported from the given validator index for the given external data ID. Result is OK if the validator reports data with zero return status, and Err otherwise.



Found an Issue?

Help us improve this page by suggesting edits on GitHub.















bandprotocol.com

This website is maintained by Band Protocol. The contents and opinions of this website are those of Band Protocol.

By using this website, you agree to our Cookie Policy.

