With Solidity

We have implemented a newStdReference interface contract to allow anyone to query data from our Standard Dataset.

To query prices from Band Protocol's oracle, a smart contract should reference Band'sStdReference contract, specifically thegetReferenceData andgetReferenceDatabulk methods.

getReferenceData

getReferenceData takes two strings (thebase andquote symbol) as the inputs, respectively. It then queries theStdReference contract for the latest rates for those two tokens, and returns aReferenceData struct, shown below.

Input

- The base symbol as typestring
- The quote symbol as typestring

Output

The base quote pair result as typeReferenceData

struct

ReferenceData

{ uint256 rate ; uint256 lastUpdatedBase ; uint256 lastUpdatedQuote ; } TheReferenceData struct has the following elements:

- Rate: the exchange rate in terms ofbase/quote
- . The value returned is multiplied by1e18
- Last updated base: the last time when the base price was updated (since UNIX epoch)
- Last updated quote: the last time when the quoted price was updated (since UNIX epoch)

Example

For example, if we wanted to query the price ofBTC/USD , the demo contract below shows how this can be done.

import interfaces / IStdReference . sol

contract

Demo

{ IStdReference public ref ;

constructor (IStdReference _ref)

public

{ ref = _ref ; }

function

demo ( )

external

view

returns

(IStdReference . ReferenceData memory)

{ return ref . getReferenceData ("BTC" ,

"USD") ; } } The result fromDemo() would yield:

ReferenceData (23131270000000000000000 ,

1659588229 ,

1659589497) Where the results can be interpreted as:

{ rate :

23131270000000000000000 ,

// 23131.27 of BTC/USD lastUpdatedBase :

1659588229 ,

// 2022-08-04 04:43:49 lastUpdatedQuote :

1659589497

// 2022-08-04 05:04:57 }

getReferenceDatabulk

The second function,getReferenceDataBulk , takes information as data arrays. For example, if you pass in['BTC','BTC','ETH'] asbase and['USD','ETH','EUR'] asquote , theReferenceDatareturned array contains the information regarding the following pairs:

- BTC/USD
- BTC/ETH
- ETH/EUR

Input

- An array of base symbols as typestring[]
- An array of quote symbol as typestring[]

Output

- An array of the base quote pair results as typeReferenceData[]

Example Contract

The following smart contract code provides some simple examples of theStdReference contract and thegetReferenceData function - these are not meant for production. TheIStdReference.sol interface definesReferenceData structure and the functions available to make the queries.

pragma

solidity

0.6 .11 ; pragma experimental ABIEncoderV2 ;

interface

IStdReference

{ /// A structure returned whenever someone requests for standard reference data. struct

ReferenceData

{ uint256 rate ;

// base/quote exchange rate, multiplied by 1e18. uint256 lastUpdatedBase ;

// UNIX epoch of the last time when base price gets updated. uint256 lastUpdatedQuote ;

// UNIX epoch of the last time when quote price gets updated. }

/// Returns the price data for the given base/quote pair. Revert if not available. function

getReferenceData (string

memory _base ,

string

```
memory _quote ) external view returns
( ReferenceData memory ) ;
/// Similar to getReferenceData, but with multiple base/quote pairs at once. function
getReferenceDataBulk ( string [ ]
memory _bases ,
string [ ]
memory _quotes ) external view returns
( ReferenceData [ ]
memory ) ; }
contract
DemoOracle
{ IStdReference ref ;
uint256
public price ;
constructor ( IStdReference _ref )
public
{ ref = _ref ; }
function
getPrice ( )
external
view
returns
( uint256 ) { IStdReference . ReferenceData memory data = ref . getReferenceData ( "BTC" , "USD" ) ; return data . rate ; }
function
getMultiPrices ( )
external
view
returns
( uint256 [ ]
memory ) { string [ ]
memory baseSymbols =
new
string [ ] ( 2 ) ; baseSymbols [ 0 ]
=
"BTC" ; baseSymbols [ 1 ]
=
"BTC" ;
```

```
string [ ]

memory quoteSymbols =

new

string [ ] ( 2 ) ; quoteSymbols [ 0 ]

=

"USD" ; quoteSymbols [ 1 ]

=

"ETH" ; IStdReference . ReferenceData [ ]

memory data = ref . getReferenceDataBulk ( baseSymbols , quoteSymbols ) ;

uint256 [ ]

memory prices =

new

uint256 [ ] ( 2 ) ; prices [ 0 ]

= data [ 0 ] . rate ; prices [ 1 ]

= data [ 1 ] . rate ;

return prices ; }

function

savePrice ( string

memory base ,

string

memory quote )

external

{ IStdReference . ReferenceData memory data = ref . getReferenceData ( base , quote ) ; price = data . rate ; } }
```

Available Reference Data Contracts

You can access theStdReference data aggregator contract on the followingsupported networks Previous Introduction Next With CosmWasm