Using the Reference Prices in Smart Contracts

To query prices from Band's oracle through smart contracts, the contract looking to use the price values should reference Band's `StdReference` contract. This contract exposes `getReferenceData` and `getReferenceDataBulk` functions.

`getReferenceData` takes two strings as the inputs, the base and quote symbol, respectively. The

latest rates for those two tokens, and returns a `ReferenceData` struct, shown below.

```
struct ReferenceData {
    uint256 rate; // base/quote exchange rate, mult
    uint256 lastUpdatedBase; // UNIX epoch of the
    uint256 lastUpdatedQuote; // UNIX epoch of the
}
```

`getReferenceDataBulk` instead takes two lists, one of the `base` tokens, and one of the `quotes`. It then proceeds to similarly queries the price for each base/quote pair at each index, and returns an array of `ReferenceData` structs.

For example, if we call `getReferenceDataBulk` with `['BTC','BTC','ETH']` and `['USD','ETH','BNB']`, the returned `ReferenceData` array will contain information regarding the pairs:

- `BTC/USD`
- `BTC/ETH`
- `ETH/BNB`

Example Usage

The contract code below demonstrates a simple usage of the new `StdReference` contract and the `getReferenceData` function.

```
pragma solidity 0.6.11;
pragma experimental ABIEncoderV2;

interface IStdReference {
    /// A structure returned whenever someone reque
    struct ReferenceData {
        uint256 rate; // base/quote exchange rate,
        uint256 lastUpdatedBase; // UNIX epoch of t
        uint256 lastUpdatedQuote; // UNIX epoch of
    }

    /// Returns the price data for the given base/q
    function getReferenceData(string memory _base,
        external
        view
        returns (ReferenceData memory);

    /// Similar to getReferenceData, but with multi
    function getReferenceDataBulk(string[] memory _
        external
        view
        returns (ReferenceData[] memory);
}

contract DemoOracle {
    IStdReference ref;
```



```
    constructor(IStdReference _ref) public {
```

```
        ref = _ref;
    }

    function getPrice() external view returns (uint
        IStdReference.ReferenceData memory data =
        return data.rate;
    }

    function getMultiPrices() external view returns
        string[] memory baseSymbols = new string[]
        baseSymbols[0] = "BTC";
        baseSymbols[1] = "BTC";

        string[] memory quoteSymbols = new string[]
        quoteSymbols[0] = "USD";
        quoteSymbols[1] = "ETH";
        IStdReference.ReferenceData[] memory data =

        uint256[] memory prices = new uint256[](2)
        prices[0] = data[0].rate;
        prices[1] = data[1].rate;

        return prices;
    }

    function savePrice(string memory base, string m
        IStdReference.ReferenceData memory data =
        price = data.rate;
    }
}
```

Using the Dataset in the Frontend