```
Title:        Price Oracles on XRP Ledger
Revision:     6 (2023-12-21)
```

Author: Gregory Tsipenyuk
Affiliation: Ripple

Price Oracles on XRP Ledger

Abstract

This proposal adds an on-chain `PriceOracle` object to the XRP Ledger. A blockchain oracle is a system or service that acts as a bridge between a blockchain network and the external world, providing off-chain data or information to decentralized applications (dApps) on the blockchain. Oracles are used to bring real-world data, for instance market prices, exchange rates, interest rates, or weather conditions onto the blockchain, enabling dApps to access and utilize information that resides outside the blockchain. This document outlines a new protocol for price oracles the on XRP Ledger, and provides guidelines for developers and system architects to implement and utilize this solution effectively. This proposal introduces a new on-ledger `PriceOracle` object and the transactions to create, delete, and update the `PriceOracle`. It also adds the `get_aggregate_price` API, to retrieve an `aggregate mean`, `trimmed mean`, and `median` for the provided price oracles. This feature requires an amendment.

Terminology

- **Oracle Provider**: A service or technology that enables the integration of external data and real-world events into a blockchain network.
- **dApp (Decentralized Application)**: An application that is built on a blockchain network and operates using smart contracts or other mechanisms or protocols for their functionality.

Creating `PriceOracle` instance on XRPL

On-Ledger Data Structures

The `PriceOracle` Object

The `PriceOracle` ledger entry represents the `PriceOracle` object on XRP Ledger and contains the following fields:

FieldName	Required?	JSON Type	Internal Type
LedgerEntryType	✓	string	UINT16
Owner	✓	string	ACCOUNTID
Provider	✓	string	BLOB
PriceDataSeries	✓	array	ARRAY
LastUpdateTime	✓	number	UINT32
URI		string	BLOB
AssetClass	✓	string	BLOB
PreviousTxnID	✓	string	HASH256
PreviousTxnLgrSeq	✓	number	UINT32

- `LedgerEntryType` identifies the type of ledger object. The proposal recommends the value 0x0080 as the reserved entry type.
- `Owner` is the account that owns this object and has the update and delete privileges. It is recommended that this account has an associated `signer list`.
- `Provider` identifies an Oracle Provider. It can be URI or any data, for instance `chainlink`. It is a string of up to 256 ASCII hex encoded characters (0x20-0x7E).
- `PriceDataSeries` is an array of up to ten `PriceData` objects, where `PriceData` represents the price information for a token pair. Any `PriceOracle` with more than five `PriceData` objects requires two owner reserves. `PriceData` includes the following fields:

FieldName	Required?	JSON Type	Internal Type
BaseAsset	✓	string	CURRENCY
QuoteAsset	✓	string	CURRENCY
AssetPrice		number	UINT64
Scale		number	UINT8

  - `BaseAsset` refers to the primary asset within a trading pair. It is the asset against which the price of the quote asset is quoted. The base asset is usually considered the 'primary' asset and forms the basis for trading. Any valid identifier, such as a stock symbol, bond CUSIP, or currency code, should be allowed and interpreted exactly like other asset identifiers in the ledger. For example, in the pair BTC/USD, BTC is the base asset; in 912810RR9/BTC, 912810RR9 is the base asset. A new type, `STI_CURRENCY`, is introduced to support the `CURRENCY` field (see Appendix for details).
  - `QuoteAsset` represents the secondary or quote asset in a trading pair. It denotes the price of one unit of the base asset. The quote asset's value is expressed in terms of the base asset. Any valid identifier such as a currency or a crypto-currency code, should be allowed and interpreted exactly like other asset

identifiers in the ledger. For example, in the pair BTC/USD, USD is the quote asset; in 912810RR9/BTC, BTC is the quote asset. A new enum value STI_CURRENCY is introduced to support the `CURRENCY` field (see Appendix for details). The `BaseAsset` and `QuoteAsset` together form a trading pair, and their relationship determines the price at which one asset can be exchanged for another.
    - `AssetPrice` is the scaled asset price, which is the price value after applying the scaling factor. This is an optional field. It is not included if the last update transaction didn't include the `BaseAsset`/`QuoteAsset` pair.
    - `Scale` is the price's scaling factor. It represents the price's precision level. For instance, if `Scale` is `6` and the original price is `0.155` then the scaled price is `155000`. Formally, $scaledPrice = originalPrice * 10^{scale}$. Valid `Scale` range is {0-10}. This is an optional field. It is not included if the last update transaction didn't include the `BaseAsset`/`QuoteAsset` pair.

- `URI` is an optional URI field to reference price data off-chain. It is limited to 256 bytes.
- `AssetClass` describes a type of the assets, for instance "currency", "commodity", "index". It is a string of up to sixteen ASCII hex encoded characters (0x20-0x7E).
- `LastUpdateTime` is the specific point in time when the data was last updated. The `LastUpdateTime` is represented as Unix Time - the number of seconds since January 1, 1970 (00:00 UTC).
- `PreviousTxnID` is the hash of the previous transaction to modify this entry (same as on other objects with this field).
- `PreviousTxnLgrSeq` is the ledger index of the ledger when this object was most recently updated/created (same as other objects with this field).

The `PriceOracle` Object ID Format

We compute the `PriceOracle` object ID as the SHA-512Half of the following values, concatenated in order:

- The Oracle space key (0x52)
- The Owner Account ID, `Owner`.
- The Oracle Document ID, `OracleDocumentID`. This field describes a unique Price Oracle instance for the given account. The Oracle Document ID is maintained by the Oracle Provider.

The `Owner` and `OracleDocumentID` uniquely identify the `PriceOracle` object and must be passed to the Oracle transactions.

Example of `PriceOracle` JSON

```
{
    "LedgerEntryType": "PriceOracle",
    "Owner": "rsA2LpzuawewSBQXkiju3YQTMzW13pAAdW",
    # "provider"
    "Provider": "70726F7669646572",
    # "currency"
    "AssetClass": "63757272656E6379",
    "PriceDataSeries": [
      {
        "PriceData": {
          "BaseAsset": "XRP",
          "QuoteAsset": "USD",
          "AssetPrice": 74,
          "Scale": 2,
        }
      },
    ],
    "LastUpdateTime": 743609414,
    "PreviousTxnID":
"C53ECF838647FA5A4C780377025FEC7999AB4182590510CA461444B207AB74A9",
    "PreviousTxnLgrSeq": 56865244
}
```

Transactions

This proposal introduces several new transactions to allow for the creation, update, and deletion of the `PriceOracle` object.

Transaction for creating or updating `PriceOracle` instance

We define a new transaction **OracleSet** for creating or updating a `PriceOracle` instance. Before the transaction can be submitted to create a new `PriceOracle` instance, the Oracle Provider has to do the following:

- Create or own the `Owner` on the XRPL with sufficient XRP balance to meet the XRP reserve and the transaction fee requirements.
- The Oracle Provider has to publish the `Owner` account public key so that it can be used for verification by dApp's.
- The Oracle Provider has to publish a registry of available Price Oracles with their unique `OracleDocumentID`. The hash of the `Owner` and the `OracleDocumentID` uniquely identifies the Price Oracle on-ledger object.

Example of OracleSet transaction JSON

```
{
    "TransactionType": "OracleSet",
    "Account": "rsA2LpzuawewSBQXkiju3YQTMzW13pAAdW",
    "OracleDocumentID": 34,
    # "provider"
    "Provider": "70726F7669646572",
    "LastUpdateTime": 743609014,
    # "currency"
    "AssetClass": "63757272656E6379",
    "PriceDataSeries": [
      {
        "PriceData": {
          "BaseAsset": "XRP",
          "QuoteAsset": "USD",
          "AssetPrice": 740,
          "Scale": 3
        }
      }
    ]
}
```

Transaction fields for OracleSet transaction

FieldName	Required?	JSON Type	Internal Type
TransactionType	✓	string	UINT16
Account	✓	string	ACCOUNTID
OracleDocumentID	✓	string	UINT32
Provider	?	string	BLOB
URI		string	BLOB
AssetClass	?	string	BLOB
LastUpdateTime	✓	number	UINT32
PriceDataSeries	✓	array	ARRAY
BaseAsset	✓	string	CURRENCY
QuoteAsset	✓	string	CURRENCY
AssetPrice	✓	number	UINT64
Scale	✓	number	UINT8

- `TransactionType` Indicates a new transaction type `OracleSet`.
- `Account` is the XRPL account that has update and delete privileges on the Oracle being set. This field corresponds to the `Owner` field on the `PriceOracle` ledger object.
- `OracleDocumentID` is a unique identifier of the Price Oracle for the given Account.
- `Provider` identifies an Oracle Provider. `Provider` must be included when creating a new instance of `PriceOracle`. It can be optionally included on update, in which case it

has to match the current `Provider` value.

- `URI` is an optional field to reference the price data off-chain.
- `AssetClass` describes the asset's type. `AssetClass` must be included when creating a new instance of `PriceOracle`. It can be optionally included on update, in which case it has to match the current `AssetClass` value.
- `LastUpdateTime` is the specific point in time when the data was last updated. `LastUpdateTime` is represented in Unix Time.
- `PriceDataSeries` is an array of up to ten `PriceData` objects, where `PriceData` represents the price information for a token pair. `PriceData` includes the following fields:
- `BaseAsset` is the asset to be priced.
- `QuoteAsset` is the denomination in which the prices are expressed.
- `AssetPrice` is the scaled asset price, which is the price value after applying the scaling factor.
- `Scale` is the price's scaling factor.

The transaction fails if:

- A required field is missing.
- XRP reserve is insufficient. If the Oracle instance has less or equal to five token pairs then the XRP reserve requirements is one, otherwise the XRP reserve requirements is two.
- Transaction's `PriceDataSeries` array size is empty or exceeds ten when creating a new Oracle instance or Oracle's instance `PriceDataSeries` array size exceeds ten after updating the Oracle instance.
- `PriceDataSeries` has duplicate token pairs.
- `PriceDataSeries` has array elements with missing `AssetPrice` and the token pair not matching an existing token pair.
- The `Account` account doesn't exist or the `Account` is not equal to the `Owner` field when updating the Oracle instance.
- The transaction is not signed by the `Account` account or the account's multi signers.
- The `URI` field length exceeds 256 bytes.
- The `Provider` field length exceeds 256 bytes.
- The `Provider` field doesn't match the current `Provider` field on update.
- The `AssetClass` field length exceeds 16 bytes.
- The `AssetClass` field doesn't match the current `AssetClass` field on update.
- The `LastUpdateTime` field is less than the previous `LastUpdateTime` or is greater than the last close time plus 30 seconds.

An `OracleSet` transaction uniquely identifies a `PriceOracle object` with its `Account` and `OracleDocumentID` fields. If such an object does not yet exist in the ledger, it is created. Otherwise, the existing object is updated. The `Provider`, `URI`, and `AssetClass` fields are copied directly from the transaction, if present. `Provider` and `AssetClass` must be included in the transaction if the object is being created.

The `PriceDataSeries` of the transaction is copied to a newly created `PriceOracle` object, or updates an existing object, like so:

- `PriceData` objects for (`BaseAsset`, `QuoteAsset`) token pairs that appear in the transaction but not the object are copied to the object.
- `PriceData` objects for token pairs that appear in both the transaction and the object are overwritten in the object.
- `PriceData` objects for token pairs that appear in both the transaction and the object and have `AssetPrice` missing in the transaction are deleted from the object.
- `PriceData` objects for token pairs that appear only in the object are left unchanged.

The order of token pairs in the transaction is not important because the token pair uniquely identifies the location of the `PriceData` object in the `PriceDataSeries` array of the `PriceOracle` object.

`PreviousTxnID`, and `PreviousTxnLgrSeq` are set in the same manner as for an `AccountSet` transaction.

The owner reserve of the account is updated according to the difference in the size of the `PriceDataSeries` before and after the transaction is applied: 0 for missing, 1 for 1 - 5 objects, 2 for 6 - 10 objects.

Transaction for deleting Oracle instance

We define a new transaction **OracleDelete** for deleting an Oracle instance.

Example of OracleDelete transaction JSON

```
{
    "TransactionType": "OracleDelete",
    "Account": "rsA2LpzuawewSBQXkiju3YQTMzW13pAAdW",
    "OracleDocumentID": 34
}
```

Transaction fields for OracleDelete transaction

FieldName	Required?	JSON Type
`TransactionType`	✓	`string`
`Account`	✓	`string`
`OracleDocumentID`	✓	`string`

- `TransactionType` indicates a new transaction type `OracleDelete`.
- `Account` is the account that has the Oracle update and delete privileges. This field corresponds to the `Owner` field on the `PriceOracle` ledger object.
- `OracleDocumentID` is a unique identifier of the Price Oracle for the given Account.

OracleDelete transaction deletes the `Oracle` object from the ledger.

The transaction fails if:

- Object with the Oracle Object ID doesn't exist.

- The `Account` account doesn't exist or the `Account` is not equal to the `Owner` field.
- The transaction is not signed by the `Account` account or the account's multi signers.

On success the transaction deletes the `Oracle` object and the owner's reserve requirement is reduced by one or two depending on the `PriceDataSeries` array size.

API's

Retrieving The Oracle

An Oracle object can be retrieved with the `ledger_entry` API call by specifying the `account` and `oracle_document_id`.

Example of `ledger_entry` API JSON

Request JSON

```
{
      "method ":  "ledger_entry ",
      "params" : [
         "oracle" : {
           "account": "rsA2LpzuawewSBQXkiju3YQTMzW13pAAdW",
           "oracle_document_id":  34,
         },
          "ledger_index ":  "validated "
      ]
}
```

Response JSON

```
{
   "index" :
"CF2C20122022DE908C4F521A96DC2C1E5EFFD1EFD47AA244E9EE9A442451162E",
   "ledger_current_index" : 23,
   "node" : {
      "Flags" : 0,
      "LastUpdateTime" : 743609014,
      "LedgerEntryType" : "Oracle",
      "Owner" : "rp847ow9WcPmnNpVHMQV5A4BF6vaL9Abm6",
      # "currency"
      "AssetClass" : "63757272656E6379",
      # "provider"
      "Provider": "70726F7669646572",
      "PreviousTxnID" :
"6F120537D0D212FEA6E11A0DCC5410AFCA95BD98D451D046832E6C4C4398164D",
      "PreviousTxnLgrSeq" : 22,
      "PriceDataSeries": [
        {
          "PriceData: {
            "QuoteAsset" : {
               "currency" : "USD"
            },
            "BaseAsset" : {
               "currency" : "XRP"
            },
            "Scale" : 1,
            "AssetPrice" : "740",
          }
        }
      ],
      "index" :
"CF2C20122022DE908C4F521A96DC2C1E5EFFD1EFD47AA244E9EE9A442451162E"
   },
   "status" : "success",
   "validated" : true
}
```

Oracle Aggregation

`get_aggregate_price` RPC calculates the aggregate price of the specified `PriceOracle` objects, and returns three types of price statistics - mean, median, and trimmed mean if `trim` parameter is included in the request. The `PriceOracle` objects are identified by the Owner Account (`account`) and Oracle Document ID (`oracle_document_id`) fields.

Example of get_aggregate_price API JSON request

```
{
"method": "get_aggregate_price",
"params": [
    {
        "ledger_index": "current",
        "base_asset": "XRP",
        "quote_asset": "USD",
        "trim": 20,
        "oracles": [
          {
            "account": "rp047ow9WcPmnNpVHMQV5A4BF6vaL9Abm6,
            "oracle_document_id": 34
          },
          {
            "account": "rp147ow9WcPmnNpVHMQV5A4BF6vaL9Abm7,
            "oracle_document_id": 56
          },
          {
            "account": "rp247ow9WcPmnNpVHMQV5A4BF6vaL9Abm8,
            "oracle_document_id": 2
          },
          {
            "account": "rp347ow9WcPmnNpVHMQV5A4BF6vaL9Abm9,
            "oracle_document_id": 7
          },
          {
            "account": "rp447ow9WcPmnNpVHMQV5A4BF6vaL9Abm0,
            "oracle_document_id": 109
          }
        ]
    }
]
}
```

Example of get_aggregate_price API JSON response

```
{
  "entire_set" : {
     "mean" : "74.75",
     "size" : 10,
     "standard_deviation" : "0.1290994448735806"
  },
  "ledger_current_index" : 25,
  "median" : "74.75",
  "status" : "success",
  "trimmed_set" : {
    "mean" : "74.75",
    "size" : 6,
    "standard_deviation" : "0.1290994448735806"
  },
  "validated" : false
  "time" : 78937648
}
```


Input API fields

FieldName	Required?	JSON Type
`ledger_index`		`string` or `number` (positive integer)
`ledger_hash`		`string`
`base_asset`	✓	`string`
`quote_asset`	✓	`string`
`oracles`	✓	`array`
`trim`		`number`
`time_threshold`		`number`

- `ledger_index` is the ledger index of the max ledger to use, or a shortcut string to choose a ledger automatically.
- `ledger_hash` is a 20-byte hex string for the max ledger version to use.
- `base_asset` is the asset to be priced.
- `quote_asset` is the denomination in which the prices are expressed.
- `oracles` is an array of `oracle` objects to aggregate over. `oracle` object has two fields:

FieldName	Required?	JSON Type
`account`	✓	`string`
`oracle_document_id`	✓	number

  - `account` is the Oracle's account.
  - `oracle_document_id` is a unique identifier of the Price Oracle for the given Account.

- `trim` is the percentage of outliers to trim. Valid trim range is 1-25. If this parameter is included then the API returns statistics for the trimmed data.
- `time_threshold` is used to define a time range in seconds for filtering out older price data. It's an optional parameter and is 0 by default; i.e. there is no filtering in this case.

The price data to aggregate is selected based on specific criteria. The most recent Price Oracle object is obtained for the specified oracles. The most recent `LastUpdateTime` among all objects is chosen as the upper time threshold. A Price Oracle object is included in the aggregation dataset if it satisfies the conditions of containing the specified `base_asset`/`quote_asset` pair, including the `AssetPrice` field, and its `LastUpdateTime` is within the time range of (upper threshold - time threshold) to the upper threshold. If a Price Oracle object doesn't contain the `AssetPrice` for the specified token pair, then up to three previous Price Oracle objects are examined and include the most recent one that fulfills the criteria.

The `get_aggregate_price` fails if:

- The oracles array size is either 0 or greater than 200.
- The oracles array's object doesn't include `account` or `oracle_document_id` or those fields have invalid value.
- `base_asset` or `quote_asset` are missing.
- `trim` or `time_threshold` contain invalid uint value.
- If the resulting data set is empty.

Output fields

On success, the response data contains the following fields:

- `entire_set` is an object of the following fields:
  - `size` is the size of the data set used to calculate the statistics.
  - `mean` is the simple mean.
  - `standard_deviation` is the standard deviation.
- `trimmed_set` is an object, which is included in the response if `trim` fields is set. The object has the following fields:
  - `size` is the size of the data set used to calculate the statistics.
  - `mean` is the simple mean.
  - `standard_deviation` is the standard deviation.
- `median` is the median.
- `time` is the most recent timestamp out of all `LastUpdateTime` values.

Appendices

Appendix 1. STI_CURRENCY

A new type, `STI_CURRENCY`, is introduced to support `BaseAsset` and `QuoteAsset` fields' type `CURRENCY`. This type can represent a standard currency code, XRP, or an arbitrary asset as a 160-bit (40 character) hexadecimal string. This type is generally conformant to the XRPL

Currency Codes. Below is a JSON example with the `BaseAsset` representing a CUSIP code `912810RR9` as a 160-bit hexadecimal string and a `QuoteAsset` representing a standard `USD` currency code:

```
{
  "PriceData" : {
    # "912810RR9"
    "BaseAsset" : "393132383130525239000000000000000000000000",
    "QuoteAsset" : "USD",
    "Scale" : 1,
    "SymbolPrice" : 740
  }
}
```