# Market data specification

Avera AI

## Contents



This doc describes an implementation of the ProcessedMarketData interface where the supplied data is represented as a dictionary of numeric values.

**IMPORTANT** The doc and the interface are under development and subject to changes.

`ProcessedMarketData` object contains the minimal data necessary for instrument pricing. It contains information about calibrated curves, volatility surfaces, and historical fixings.

`MarketDataDict` is an implementation of `ProcessedMarketData` interface where users are expected to supply market data as a nested dictionary in the following format:

```
{
  # Currencies
  'rates': {
    'USD': {
      'curves': {
        'risk_free_curve': {
           # Sorted tensor of dates
          'dates': [date1, date2]
          # Discount factors
          'discounts': [discount1, discount2]
          # Interpolation configuration
          'config': curve_config
        }
        'LIBOR_3M': {
           # Sorted tensor of discount dates
          'dates': [date1, date2]
          # Discount factors
```

```
          'discounts': [discount1, discount2]
          # Fixing source
          'fixings_source': 'LIBOR_3M'
          # Interpolation configuration
          'config': curve_config
        }
      }
      'historical_fixings': {
        'LIBOR_3M': {
          # Sorted tensor of fixing dates
          'fixing_dates': [fixing_date1, fixing_date2,
                           fixing_date3]
          # Fixing rates represented as annualized simple rates
          'fixing_rates': [rate1, rate2, rate3]
          # Fixing daycount. Should have a value that is one of
          # the `core.DayCountConventions`. Must be supplied if
          # the fixings are specified
          'fixing_daycount': 'ACTUAL_360'
          # Business day convention
          'business_convention': 'FOLLOWING'
          # Holiday calendar
          'holiday_calendar': 'US'
        }
      }
      'volatility_cube': {
        'SWAPTION': {
        # Option expiry
        "expriries": [date1, date2]
        # Swap tenors for each expiry. In each row, the last value
        # can be repeated
        "tenors": 2d array
        # Strikes for each `tenors` entry.
        "strikes": 3d array
        # Implied volatilities corresponding to each `strikes` entry
        "implied_volatilities": 3d array
        # Interpolation configuration
        'config': volatility_config
        }
      'CAP': {
       ....
       }
      }
    }
  }
  # Equities
```

```
  'equities':{
    "USD": {
      "GOOG": {
        "spot_price": 1700
        "historical_spot_prices": {
          "dates": [date1, date2]
          "values": [price1, price2]
        }
        "currency": "USD"
        "volatility_surface": {
          "dates": [date1, date2]
          # Strikes for each date. In each row, the last value can
          # be repeated, e.g,
          # [[1650, 1700, 1750, 1800, 1800],
          #  [1650, 1700, 1750, 1770, 1790]]
          "strikes": 2d array
          # Implied volatilities corresponding to the dates and
          # strikes.
          "implied_volatilities": 2d array
          # Interpolation configuration
          'config': volatility_config
        }
      }
    }
  }
# Snapshot date
"reference_date": date
}
```

Each top-level key of the dictionary refers to a member of an asset class. This can be a rate, equity, commodity, etc. Above only rates and equities are included. `reference_date` refers to the snapshot date.

Below we specify rates and equity classes specifications.

## Rates

- Top level of a rate object corresponds to a currency. For example, `USD` above. Each currency has associated rate curves (e.g., LIBOR above), fixings and volatility cubes.
- `curves` should contain a dictionary of curve specification that are needed for pricing. Available keys are the proto name fields of `RateIndexType`. Additionally, `risk_free_curve` should be supplied, which is the default discounting curve. Curves are represented via discount factors and corresponding `dates` (**TODO: add relatives dates `rdates` alternative**).

In order to add a missing curve type, simply add it as a new field in `RateIndexType`

- Fixings can be supplied for each curve using `fixing_rates`, `fixing_dates`, and `fixing_daycount` fields (available day count conventions are listed in the `DayCountConvention` proto). If the fields are not supplied, fixings are assumed to be equal to zero. Otherwise, fixings are treated as a piecewise constant function for the dates that are not supplied with `fixing_dates`

Additional curve configuration (e.g., day count convention interpolation method, etc) and customization should be supplied via the configuration dictionaries `curve_config` and `volatility_config` (see Market data config section below).

- `volatility_cube` contains volatility cube information. There can be different cube types, e.g., 'SWAPTION' or 'CAP' above (**TODO: standardize volatility cube types**). Each cube consists of a vector of option expiries an array of `tenors`, and a cube of `strikes` and the corresponding `implied_volatilities`.

## Equities

Top level contains `currency` name which gather all associated equities. (e.g., `GOOG` above). Users have to specify `spot_price`, `volatility_surface` (represented as `strikes`, `dates` and `implied_volatilities`) and, if needed, `historical_spots` data.

## Market data config

Market data configuration allows users to specify the interpolation methods and day count conventions. Currently the following format is supported

```
curve_config = {"interpolation_method": "LINEAR",
                "interpolate_rates": False,
                "daycount_convention": "ACTUAL_365",
                # Business day convention
                'business_convention': 'FOLLOWING',
                # Holiday calendar
                'holiday_calendar': 'US'}
```

where `interpolation_method` is one of the supported methods, `interpolate_rates` is a boolean that specifies whether the interpolation is performed in a rates or discount factor space, `daycount_convention` is one of the supported conventions, `business_convention` is one of BusinessDayConvention, and `holiday_calendar` is a BankHoliday.

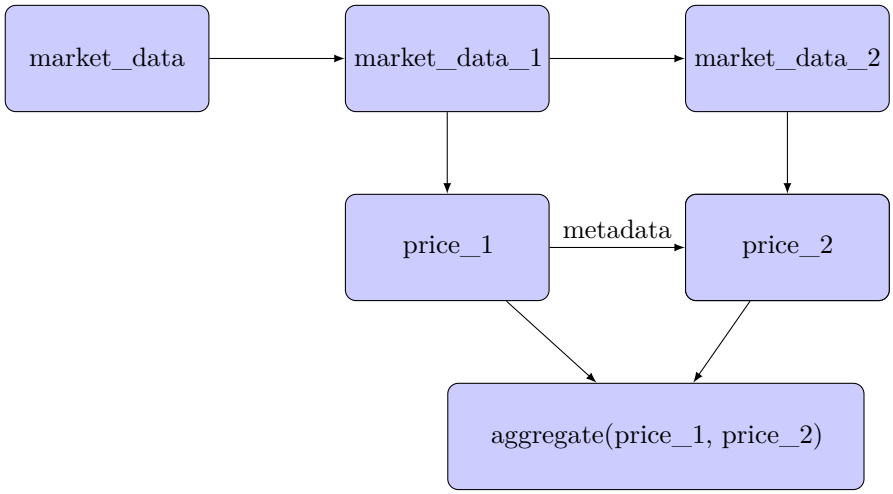
By default cubic interpolation is used in the discount factor space with `ACTUAL_365` day count convention.

For the volatility surface/cube day count, business days and calendar specifications are supported.

```
curve_config = {"daycount_convention": "ACTUAL_365",
                # Business day convention
                'business_convention': 'FOLLOWING',
                # Holiday calendar
                'holiday_calendar': 'US'}
```

## Fixings

Fixing information for rates/spots should be provided either via the market data or via the instrument definition. The fixing information can be updated via metadata supplied at the instrument pricing stage. For example, consider the following diagram




Here market data is evolved twice and the pricing is performed at each stage. Fixings can either be a part of the market data or supplied separately via pricing config. For example, config may contain a knock-out event for a barrier option. Metadata is used by the pricing method via the configuration field

```
Instrument.from_protos(proto_list,
                       config=InstrumentConfig(**metadata))
```

By default, the pricing method should check if instrument metadata contains the requested fixing data and, otherwise, use market data to extract the fixings.