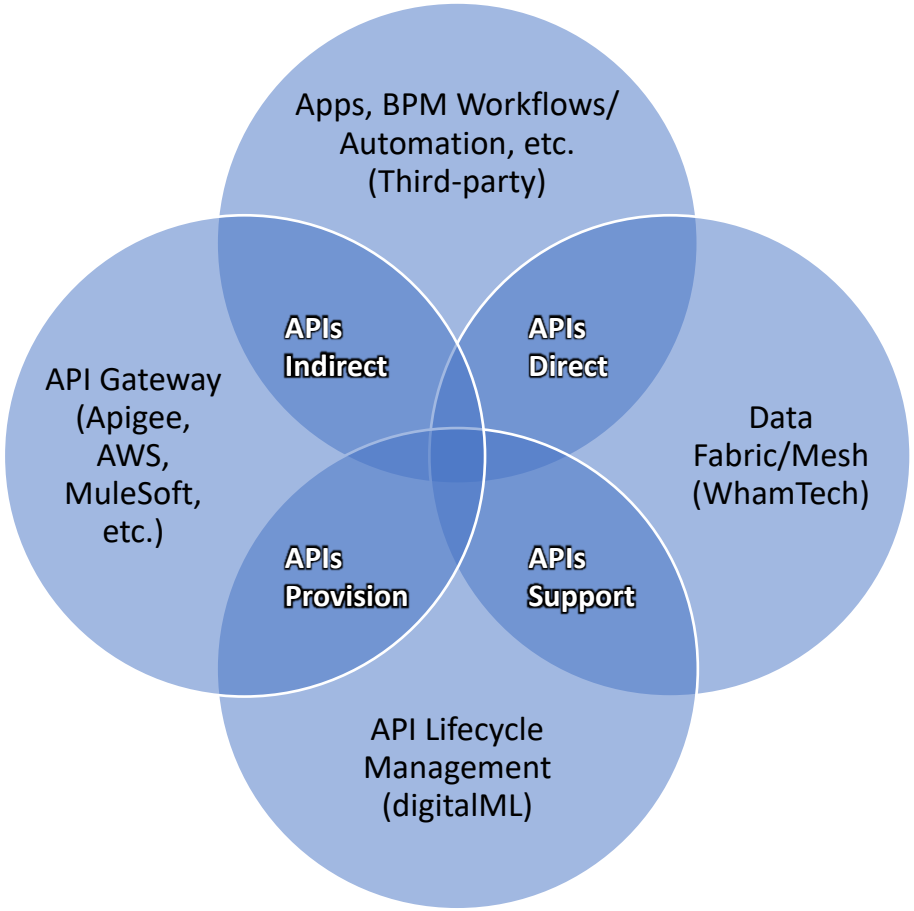
Open Banking APIs Solution GetPrice Microservices POC

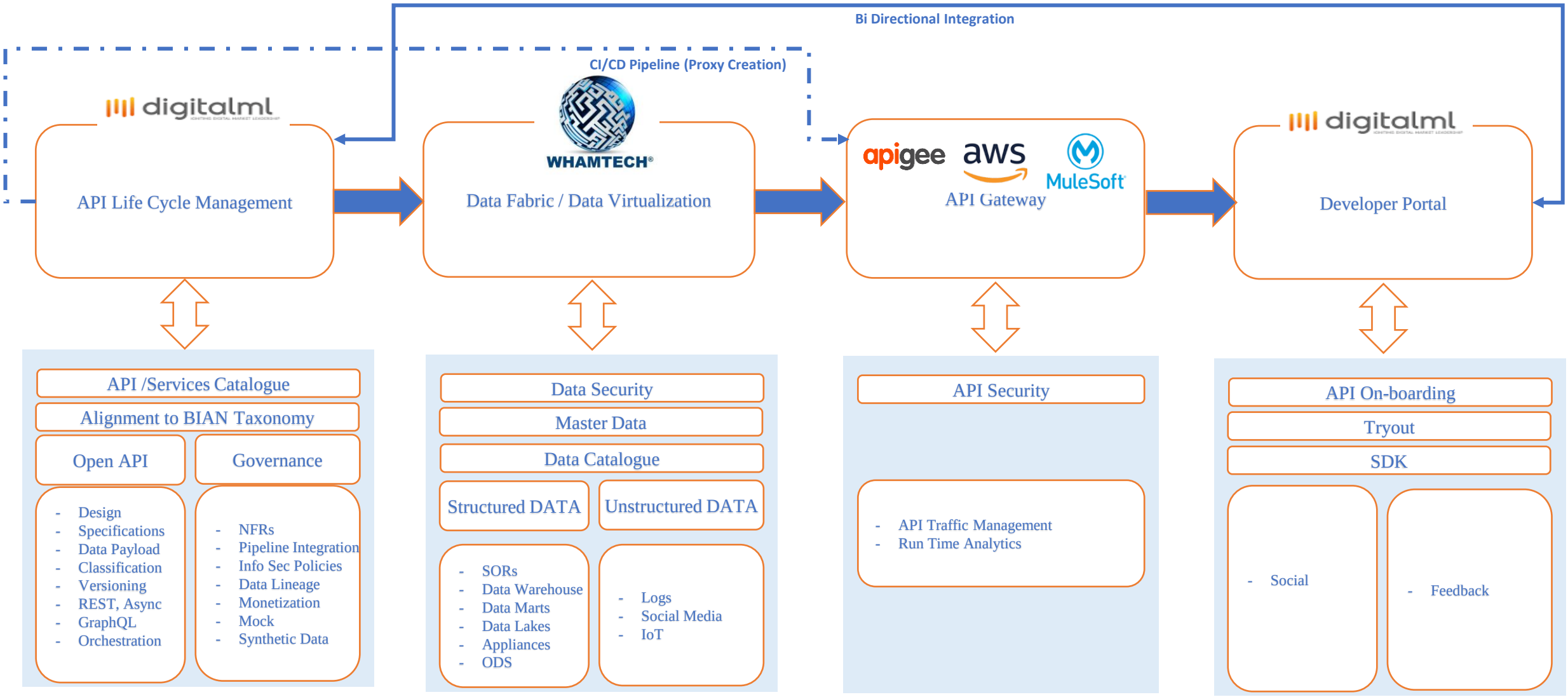
September 1, 2022

Revision 1.3

Microservices and APIs-based combination for vertical market-specific solutions



Data Fabric/Mesh Solution Accelerator



Some assumptions/thoughts

- For a good definition of microservices and APIs: https://hardiks.medium.com/how-are-microservices-different-from-apis-614e5c02e7e8
- GetPrice Rest API to use Ticker or list of Tickers, and Date to query pricing tables for latest price for the given date, followed by additional steps, consisting of a total of 4 Rest APIs and 2 input and response Rest APIs
- Apply business logic listed on slide 5* in GetPrice Rest API
- Blue box process on slide 6 to follow input of Ticker or list of Tickers, and Date
- Create an orchestration layer using RabbitMQ to manage the process

Domain-Driven Design (DDD) Artifacts (for more info, see https://www.digitalml.com/)

- ✓ Value Object
- ✓ Entity
- ✓ Aggregate
- ✓ Aggregate Root
- ✓ Repository
- ✓ Service Object
- ✓ Service
- ✓ Domain Event
- ✓ Bounded Context

Request
- Send Date
- Request (send List Tickers) to Get Price
- Send specific Ticker

Response
- List of Instrument with valid Prices
- Send Specific Instrument with Valid Price

Payload
- Instrument ID (1234)
- Instrument Type (EQ)
- Instrument Currency (USD)
- Exchange Code (NYC)
- Country
- Source (Bloomberg)
- Ticker (AAPL)
- Date (07/05/22)
- Price Indicator (NYC Close)
- Price ($109.56)

Format
- Json
- CSV
- XML
- YAML

Error Handling
- **200, 300, 400**

Service Functionality:
- Ability to Price Single Instrument
- Ability to Price List of Instruments
- Business Logic*: (Price Selection Logic)
  - When returning the price, always use Bloomberg Price
  - If Bloomberg Price does not exist (Null or zero), use Reuters

NOTE: Bloomberg and Reuters do not know Bank's Internal Instrument IDs.
So, when sending Request to Price, send them Ticker(s) OR (ISIN, SEDOL, CUSIP, Exchange Code, Country, Currency).

Uniqueness of Instrument: Any combination of:
- ISIN/CUSIP/SEDOL, Exchange Code, Country, Currency

Instrument
- Instrument ID (1234)
- ISIN
- SEDOL
- CUSIP
- Exchange Code
- Country
- Currency
- Ticker
- Instrument Type
- Instrument Name
- Date
- Price

PriceHistory
- Instrument ID (1234)
- Date
- Price

BloombergPrice
- ISIN
- CUSIP
- Exchange Code
- Country
- Currency
- Ticker
- Instrument Type
- Price
- Date

ReutersPrice
- ISIN
- CUSIP
- Exchange Code
- Country
- Currency
- Ticker
- Instrument Type
- Price
- Date

Demo Files:
DemoInstrument.csv (Contain 100 Instruments with CUSIPS)

1 Instrument ID n

Instrument
- **Instrument_Id(1234)**
- ISIN
- CUSIP
- Exchange_Cd
- Country_Cd
- Currency_Cd
- Ticker
- Instrument_Type
- Instrument_Name
- Pricing_Source
- Instrument_Price
- Pricing_Date
- Row_Insert_Time
- Row_update_time
- Row_Updated_By

Business Key

Instrument ID – Unique Key

PriceHistory
- **Instrument ID (1234)**
- Price
- Date
- Pricing_Source

BloombergPrice
- **ISIN**
- **CUSIP**
- **Exchange Code**
- **Country**
- **Currency**
- Ticker
- Instrument Type
- Price
- Date

Generated(using Yahoo Finance) **BloomberJun.csv**

ReutersPrice
- **ISIN**
- **CUSIP**
- **Exchange Code**
- **Country**
- **Currency**
- Ticker
- Instrument Type
- Price
- Date

Generated(using Yahoo Finance) **ReutersJun.csv**

Instrument.Prices **Price.Instrument**

DDD – These are defined as collections

Database Prep:

- Load Instrument Table into Database
- Load BloombergPrice Table
- Load ReutersPrice Table

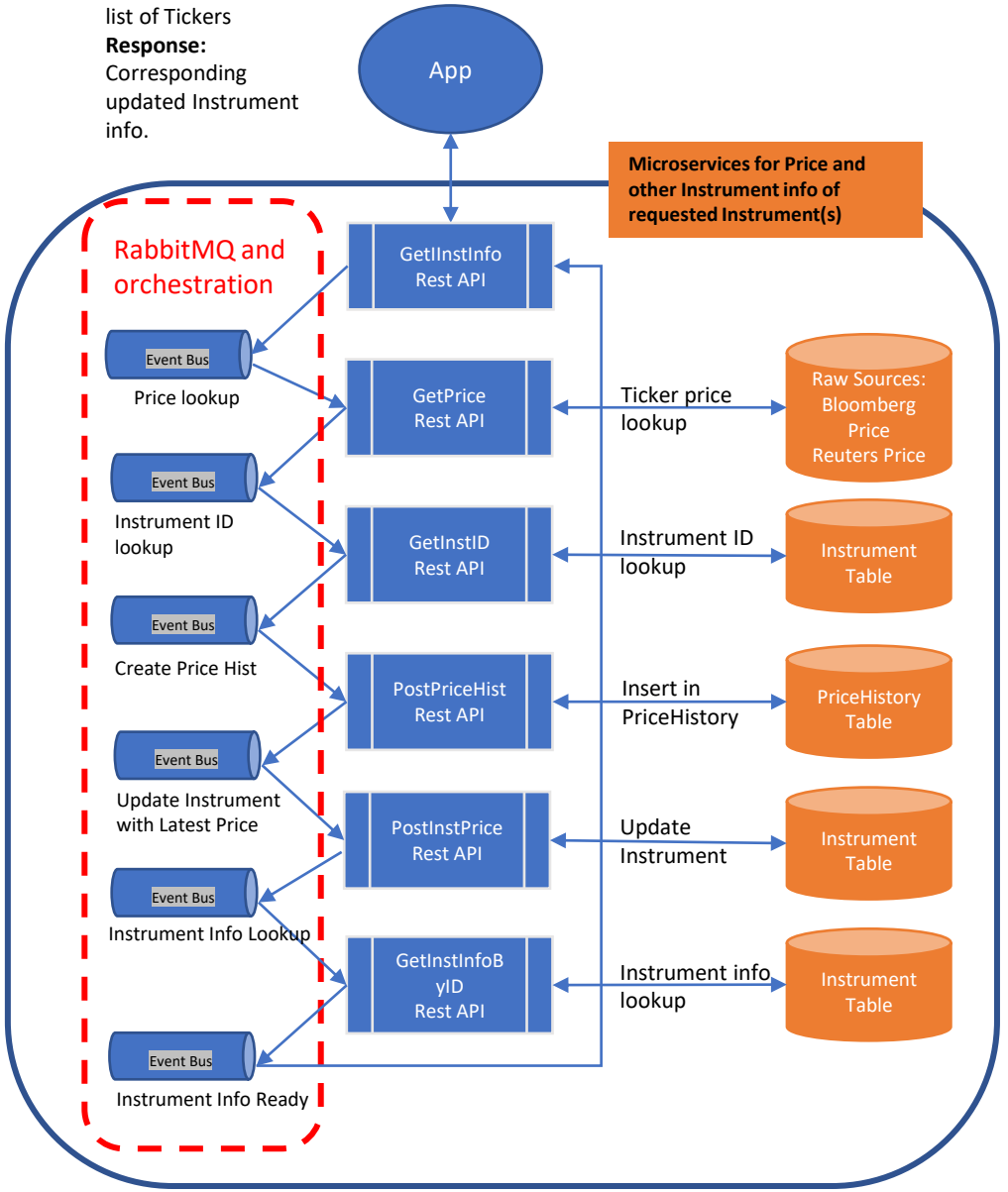
DDD Artifacts to be created on Ignite:

- ✓ **Value Object**: ISIN, CUSIP, Exchange Code, Country, Currency, Instrument Type, Ticker, Source
- ✓ **Entity**: Instrument, PriceHistory, Bloomberg, Reuters
- ✓ **Aggregates**: BloomberPrice, ReutersPrice, PriceHistory, Instrument
- ✓ **Aggregate Root:** Instrument
- ✓ **Repository**: GetPrice, GetInstID
- ✓ Service Object
- ✓ **Service**: PostPriceHist, PostInstPrice
- ✓ **Domain Event**: Event to find Instrument ID based on value objects
- ✓ **Bounded Context:** Trade & Price Reporting; Sub-Domain:Pricing

DDD Rules:

- Aggregate should not cross deployment Boundary

Input: Date, Ticker or list of Tickers
Response: Corresponding updated Instrument info.



Link to POC demo

Below is the Swagger link to ProcessInstInfo() API. This API is a variation of GetInstInfo() high-level API and shows the internal orchestration trace. It takes a date and comma separated ticker list as input and returns the price info and the trace.

The RabbitMQ-based orchestration logic calls internal APIs sequentially as described in the previous slide. Currently, the orchestration logic is within the API code that can be defined/modified by a business person in a file that could be configured through a BPM tool. In the future, the orchestration logic can be defined/modified via a text input file that describes the orchestration logic in simple business terms (no coding needed) or could be configured through a BPM tool.

http://18.216.221.170:8080/swagger-ui/index.html#/Pricing%20Services/processInstrumentsGetUsingGET

This is the main link to the APIs:

http://18.216.221.170:8080/swagger-ui/index.html

Swagger UI

Not secure | 18.216.221.170:8080/swagger-ui/index.html#/Pricing%20Services/processInstrumentsGetUsingGET

Swagger Supported by SMARTBEAR

Select a definition: default

WhamTech OpenBanking REST API 1.0

[Base URL: 18.216.221.170:8080/]
http://18.216.221.170:8080/v2/api-docs

REST APIs to access WhamTech Microservices for Stock Prices

WhamTech Development Team - Website
Send email to WhamTech Development Team
Proprietary License

Pricing Services Pricing Controller

POST **/GetInstId** Internal API to Lookup instrument id for the ticker(s)

POST **/GetInstInfo** High Level API to orchestrate the pricing of tickers

POST **/GetPrice** Internal API to Get the best price for ticker(s) and request date

POST **/PostInstPrice** Internal API to Update the instrument details to instrument table

POST **/PostPriceHist** Internal API to Insert/Update the price details to price history table.

GET **/ProcessInstInfo** High Level API to orchestrate the pricing of tickers

This Microservice runs the orchestration using RabbitMQ which inserts/updates price request information in Instrument and PriceHistory tables from Bloomberg and Reuters sources. A sample request will have date: 2022-06-02 and tickerList: GE, AAPL

Parameters Try it out

Name	Description

Click to add notes

Slide 5 of 5 Accessibility: Investigate Notes Comments 107%

Swagger UI

Not secure | 18.216.221.170:8080/swagger-ui/index.html#/Pricing%20Services/processInstrumentsGetUsingGET

Parameters

Try it out

Name	Description
dateOfRequest * required string *(query)*	Sample Date: 2022-06-02 dateOfRequest - Sample Date: 2022-06-02
tickerList * required string *(query)*	Sample list of tickers: AAPL,GE tickerList - Sample list of tickers: AAPL,GE

Responses

Response content type: application/json

Code	Description
200	OK Example Value | Model

```
{
  "requestId": "string",
  "status": "string",
  "updatedStockPrices": [
    {
      "stockPrice": 0,
      "stockPriceDate": 0,
      "stockSource": "string",
      "ticker": "string"
    }
  ]
}
```

Models

Click to add notes

Slide 5 of 5 | Accessibility: Investigate | Notes | Comments | 107%

Swagger UI

Not secure | 18.216.221.170:8080/swagger-ui/index.html#/Pricing%20Services/processInstrumentsGetUsingGET

This Microservice runs the orchestration using RabbitMQ which inserts/updates price request information in Instrument and PriceHistory tables from Bloomberg and Reuters sources. A sample request will have date: 2022-06-02 and tickerList: GE, AAPL

Parameters Cancel

Name	Description
dateOfRequest * required string *(query)*	Sample Date: 2022-06-02 2022-06-02
tickerList * required string *(query)*	Sample list of tickers: AAPL,GE AAPL, GE

Execute

Responses Response content type application/json

Code	Description
200	OK Example Value | Model

```
{
  "requestId": "string",
  "status": "string",
  "updatedStockPrices": [
    {
      "stockPrice": 0,
      "stockPriceDate": 0,
      "stockSource": "string",
      "ticker": "string"
    }
  ]
}
```

Click to add notes

Slide 5 of 5 Accessibility: Investigate Notes Comments 107%

Swagger UI

Not secure | 18.216.221.170:8080/swagger-ui/index.html#/Pricing%20Services/processInstrumentsGetUsingGET

Execute | Clear

Responses Response content type: application/json

Curl

```
curl -X GET "http://18.216.221.170:8080/ProcessInstInfo?dateOfRequest=2022-06-02&tickerList=AAPL%2C%20GE" -H "accept: application/json"
```

Request URL

```
http://18.216.221.170:8080/ProcessInstInfo?dateOfRequest=2022-06-02&tickerList=AAPL%2C%20GE
```

Server response

Code	Details
200	Response body


```
{
  "status": "Done",
  "updatedStockPrices": [
    {
      "ticker": "AAPL",
      "stockPrice": 147.7300018,
      "stockSource": "Reuters",
      "stockPriceDate": "2022-06-01T11:00:00.000-05:00"
    },
    {
      "ticker": "GE",
      "stockPrice": 77.82694759,
      "stockSource": "Bloomberg",
      "stockPriceDate": "2022-06-01T11:00:00.000-05:00"
    }
  ],
  "traceString": [
    "2022-08-31 16:03:03.208: In REST Call: /GetInstInfo, sending event to Queue: PriceLookup()",
    "2022-08-31 16:03:03.209: Queue: PriceLookup(), invoking REST endpoint: /GetPrice",
    "2022-08-31 16:03:03.230: Queue: PriceLookup(), sending event to Queue: InstLookup()",
    "2022-08-31 16:03:03.232: Queue: InstLookup(), invoking REST endpoint: /GetInstId",
    "2022-08-31 16:03:03.239: Queue: InstLookup(), sending event to Queue: CreatePriceHist()",
    "2022-08-31 16:03:03.241: Queue: CreatePriceHist(), invoking REST endpoint: /PostPriceHist",
    "2022-08-31 16:03:03.252: Queue: CreatePriceHist(), sending event to Queue: UpdateInst()",
    "2022-08-31 16:03:03.254: Queue: UpdateInst(), invoking REST endpoint: /PostInstPrice",
    "2022-08-31 16:03:03.289: Queue: UpdateInst(), sending event to Queue: InstInfoReady()",
    "2022-08-31 16:03:03.291: Out REST Call: /GetInstInfo, Returning response."
  ]
}
```


Download

Response headers

Click to add notes

Slide 5 of 5 | Accessibility: Investigate | Notes | Comments | 107%

```
  "traceString": [
    "2022-08-31 16:03:03.208: In REST Call: /GetInstInfo, sending event to Queue: PriceLookup()",
    "2022-08-31 16:03:03.209: Queue: PriceLookup(), invoking REST endpoint: /GetPrice",
    "2022-08-31 16:03:03.230: Queue: PriceLookup(), sending event to Queue: InstLookup()",
    "2022-08-31 16:03:03.232: Queue: InstLookup(), invoking REST endpoint: /GetInstId",
    "2022-08-31 16:03:03.239: Queue: InstLookup(), sending event to Queue: CreatePriceHist()",
    "2022-08-31 16:03:03.241: Queue: CreatePriceHist(), invoking REST endpoint: /PostPriceHist",
    "2022-08-31 16:03:03.252: Queue: CreatePriceHist(), sending event to Queue: UpdateInst()",
    "2022-08-31 16:03:03.254: Queue: UpdateInst(), invoking REST endpoint: /PostInstPrice",
    "2022-08-31 16:03:03.289: Queue: UpdateInst(), sending event to Queue: InstInfoReady()",
    "2022-08-31 16:03:03.291: Out REST Call: /GetInstInfo, Returning response."
  ]
}
```


Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Wed31 Aug 2022 16:03:03 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description
200	OK

Example Value | Model


```
{
  "requestId": "string",
  "status": "string",
  "updatedStockPrices": [
    {
      "stockPrice": 0,
      "stockPriceDate": 0,
      "stockSource": "string",
      "ticker": "string"
    }
  ]
}
```


Models

RequestStockInstrument {
requestId string
stockPriceList [StockPrice > {...}]
}

Click to add notes

Models

RequestStockInstrument {
- requestId : string
- stockPriceList : [StockPrice > {...}]

}

RequestStockPrice {
- dateOfRequest : string($date)
- tickerList : [string]

}

RequestStockTrackPrice {
- stockPriceList : [StockTrackPrice > {...}]

}

ResponseInstrument {
- requestId : string
- status : string
- updatedStockPrices : [StockPrice > {...}]

}

ResponseStockInstrument {
- stockInstrument : [StockInstrument > {...}]

}

ResponseStockPrice {
- dateOfRequest : string($date)
- stockPrice : [StockPrice > {...}]

}

StockInstrument {

ResponseInstrument {
- requestId: string
- status: string
- updatedStockPrices: [StockPrice {...}]

}

ResponseStockInstrument {
- stockInstrument: [StockInstrument {...}]

}

ResponseStockPrice {
- dateOfRequest: string($date)
- stockPrice: [StockPrice {...}]

}

StockInstrument {
- id: integer($int64)
- stockPrice: number
- stockPriceDate: integer($int64)
- stockSource: string
- ticker: string

}

StockPrice {
- stockPrice: number
- stockPriceDate: integer($int64)
- stockSource: string
- ticker: string

}

StockTrackPrice {
- id: integer($int64)
- stockPrice: number
- stockPriceDate: integer($int64)
- stockSource: string

}