Contents

Blpapi
  Introduction ................................................. 1
  The API .................................................... 1
    Create a session ........................................ 2
    Reference data request ................................. 2
    Historical Data ......................................... 3
    Intraday Tick Data ...................................... 6
    Intraday Bar Data ....................................... 8

Blpapi

Introduction

Blpapi.jl is the Julia package for connecting to Bloomberg using the BLPAPI C client.

The package is supplied as an installable package that can be installed on top of a JuliaPro installation. The installer arranges the paths based on the location of JuliaPro. The Bloomberg client-side libraries are installed as part of this package.

A running Bloomberg terminal is required. This will be running on the same PC as Julia and this code. This library will connect to the Bloomberg terminal over a local socket, and fetch data using the blpapi client libraries.

The API

using Blpapi

There are four main exported functions bdp for reference data, bdh for historical data, tick for tick data, and bar for bar data. The names are inspired by the Bloomberg excel connector.
Create a session

Create a Bloomberg session which will connect you to the Bloomberg terminal by providing IP address and port number

IP = "localhost"
Port = 8194
session = createSession(IP, Port)

Reference data request

#required parameters of reference data request:
#ticker names
tickers = ["IBM US Equity", "AAPL US Equity"]

#fields requested
fields = ["PX_Open", "PX_High", "PX_Last"]

#Call the bdp function by providing session, tickers and fields variables
# getting the response in variable ‘Response’
Response = bdp(Session, tickers, fields)

The response from bdp function is a Julia type ReferenceDataResponse object. Individual field values can be obtained by indexing this object.

# extracting response data by providing ticker and field name
ibmLastPrice = Response["IBM US Equity", "PX_Last"]
ibmOpenPrice = Response["IBM US Equity", "PX_Open"]
appleLastPrice = Response["AAPL US Equity", "PX_Last"]
appleOpenPrice = Response["AAPL US Equity", "PX_Open"]

NOTE: Whenever the data is not available for provided ticker and field in the Response, NullException() will be thrown

Value = Response["IBM US Equity", "PX_High"]
Value = Response["Foo Bar", "PX_Last"]
Value = Response["Foo", "Bar"]

All the above attempts will throw NullException()

Optional Parameters:

returnFormattedValue: Setting this to 1 will force all data to be returned as a string

bdp( { Fixed Parameters } ; returnFormattedValue = 1)

useUTCTime: Setting this to 1 returns values in UTC
bdp( { Fixed Parameters } ; useUTCTime = 1)

forcedDelay: Setting to 1 will return the latest data up to the delay period specified by the exchange for this security.

bdp( { Fixed Parameters } ; forcedDelay = 1)

fieldID: Field mnemonic, PRICING_SOURCE, or field alpha-numeric value: Desired override value. This along with fieldID is used to append overrides to modify the calculation.

bdp( { Fixed Parameters } ; fieldID = "PRICING_SOURCE", value = "CG")

Multiple optional parameters can be provided in a request.

bdp(session, tickers, fields' returnFormattedValue = 1, useUTCTime = 1)

Historical Data

#Fixed Parameters of Historical Data request:
#ticker names
tickers = ["IBM US Equity", "AAPL US Equity"]

#fields requested
fields = ["PX_Open", "PX_High", "PX_Last"]

#startDate in the YYYYMMDD format
startDate = "20150601"

#endDate in the YYYYMMDD format you want to send request till:
endDate = "20150701"

#Calling the function with fixed parameters:
#Call the bdh function by providing session, 
# ticker, fields, startDate and endDate variables
bdh(session, tickers, fields, startDate, endDate)

The response from bdh function is a Julia type HistoricalDataResponse object.

# initializing tickers and fields array to be passed in bdh function
tickers = ["IBM US Equity", "AAPL US Equity"]
fields = ["PX_Last", "PX_Open"]

# initializing startDate and endDate variables
startDate = "20150601"
endDate = "20150701"
The API

# getting the response in variable 'Response'
Response = bdh(Session, tickers, fields, startDate, endDate)

# extracting response data by providing ticker, field name and date string
# the date string to be passed to Response must be in YYYYY-MM-DD format
ibmLastPrice = Response["IBM US Equity", "PX_Last", "2015-07-01"]
ibmOpenPrice = Response["IBM US Equity", "PX_Open", "2015-06-24"]
apleLastPrice = Response["AAPL US Equity", "PX_Last", "2015-06-24"]
apleOpenPrice = Response["AAPL US Equity", "PX_Open", "2015-06-10"]

# extracting all response data for a particular field
Response["IBM US Equity", "PX_Last"]

NOTE: Whenever the data is not available for provided ticker, field and date in the Response, NullException() will be thrown
Value = Response["IBM US Equity", "PX_Last", "1999-01-01"]
Value = Response["Bla Bla", "PX_Last", "2015-07-01"]
Value = Response["Bla Bla", "Bla Bla", "2015-07-01"]

All the above attempts will throw NullException()

Optional Parameters

periodicitySelection: Determine the frequency of the output. To be used in conjunction with periodicityAdjustment. Can have any one of the values from DAILY / WEEKLY / MONTHLY / QUARTERLY / SEMI_ANNUALLY / YEARLY. Default value of this option is WEEKLY

bdh( { Fixed Parameters } ; periodicitySelection = "MONTHLY")

periodicityAdjustment: Determine the frequency and calendar type of the output. To be used in conjunction with periodicitySelection. Can have any one of the values from ACTUAL / CALENDAR / FISCAL. Default value of this option is ACTUAL

bdh( { Fixed Parameters } ; periodicityAdjustment = "ACTUAL")

currency: This is the three letter ISO code. Amends the value from local to desired currency

bdh( { Fixed Parameters } ; currency = "GBP")

adjustmentNormal: Setting this to 1 will adjust historical pricing to reflect: Regular Cash, Interim, 1st Interim, 2nd Interim, 3rd Interim, 4th Interim, 5th Interim, Income, Estimated, Partnership Distribution, Final, Interest on Capital, Distribution, Prorated
adjustmentAbnormal: Setting this to 1 will adjust historical pricing to reflect: Special Cash, Liquidation, Capital Gains, Long-Term Capital Gains, Short-Term Capital Gains, Memorial, Return of Capital, Rights Redemption, Miscellaneous, Return Premium, Preferred Rights Redemption, Proceeds/Rights, Proceeds/Shares, Proceeds/ Warrants

adjustmentSplit: Setting this to 1 will adjust historical pricing and/or volume to reflect: Special Cash, Liquidation, Capital Gains, Long-Term Capital Gains, Short-Term Capital Gains, Memorial, Return of Capital, Rights Redemption, Miscellaneous, Return Premium, Preferred Rights Redemption, Proceeds/Rights, Proceeds/Shares, Proceeds/ Warrants

overrideOption: Indicates whether to use the average or the closing price in quote calculation. Can have any one of these values: OVERRIDE_OPTION_CLOSE and OVERRIDE_OPTION_GPA

pricingOption: Sets quote to Price or Yield for a debt instrument whose default value is quoted in yield (depending on pricing source). Can have any one of these values: PRICING_OPTION_PRICE and PRICING_OPTION_YIELD

nonTradingDayFillOption: Sets to include/exclude non trading days where no data was generated. Can have any one of these values; NON_TRADING_WEEKDAYS, ALLCALENDAR_DAYS and ACTIVE_DAYS_ONLY

calendarCodeOverride: Returns the data based on the calendar of the specified country, exchange, or religion. Taking a two character calendar code, returns the data based on the calendar of the specified country, exchange, or religion. NOTE: Can only be used when periodicitySelection is DAILY

maxDataPoints: The response will contain up to X data points, where X is the integer specified. If the original data set is larger than X, the response will be a subset, containing the last X data points.
fieldID: Field mnemonic, PRICING_SOURCE, or field alpha-numeric value: Desired override value. This along with fieldID is used to append overrides to modify the calculation.

```
bdh( { Fixed Parameters } ; fieldID = "PRICING_SOURCE", value = "CG")
```

Multiple optional parameters can be passed to a request

```
bdh(session, tickers, fields, startDate, endDate, periodicitySelection = "MONTHLY", periodicityAdjustment = "ACTUAL")
```

Intraday Tick Data

```
#Fixed Parameters of Intraday Tick Data request:
ticker = "IBM US Equity"

eventTypes = ["TRADE", "BID"]

startDateTime = "2015-10-27T15:55:00"

endDateTime = "2015-10-27T16:00:00"

#Calling the function with fixed parameters:
tick(session, ticker, eventTypes, startDateTime, endDateTime)
```

The response from tick function is a Julia object of type TickDataResponse. TickDataResponse contains many response elements of type TickDataElement. Each TickDataElement has following variables:

- valueVar (Dependent on eventType passed)
- sizeVar (Integer)
- conditionCodeVar (ASCIIString)
- exchangeCodeVar (ASCIIString)
- micCodeVar (ASCIIString)
- brokerBuyCodeVar (ASCIIString)
- brokerSellCodeVar (ASCIIString)
- rpsCodeVar (ASCIIString)

Default values for these variables is a string with length zero. This indicates that response don’t have data for those variables.

```
# initializing ticker and eventTypes array to be passed in tick function
ticker = "IBM US Equity"
eventTypes = ["TRADE", "BID"]
```
# initializing startDateTime and endDateTime variables
startDateTime = "2015-10-27T15:55:00"
endDateTime = "2015-10-27T16:00:00"

# getting the response in variable ‘Response’
Response = tick(Session, ticker, eventTypes, startDateTime, endDateTime)

# extracting response elements from variable ‘Response’.
responseElementTrade = Response["TRADE", DateTime(2015, 10, 27, 15, 55)]
responseElementBID = Response["BID", DateTime(2015, 10, 27, 15, 57)]

# extracting data from response elements
tradeValue = responseElementTrade.valueVar
bidValue = responseElementBID.valueVar

NOTE: Whenever the data is not available for provided eventType and DateTime object in the Response, NullException() will be thrown, when extracting response elements.

Value = Response["TRADE", DateTime(1999, 10, 27, 15, 55)]
'Value = Response["Bla Bla", DateTime(2015, 10, 27, 15, 55)]

All the above attempts will throw NullException()

Optional Parameters:

returnEids: Setting this to 1 will return the entitlement identifiers (EIDs) associated with security

tick( { Fixed Parameters } ;  returnEids = 1)

includeConditionCodes: Setting this to 1 will return any condition codes that may be associated to a tick, which identifies extraordinary trading and quoting circumstances

tick( { Fixed Parameters } ;  includeConditionCodes = 1)

includeNonPlottableEvents: Setting this to 1 will return all ticks, including those with condition codes

tick( { Fixed Parameters } ;  includeNonPlottableEvents = 1)

includeExchangeCodes: Setting this to 1 will return the exchange code of the trade

tick( { Fixed Parameters } ;  includeExchangeCodes = 1)

includeBrokerCodes: Setting this to 1 will return the broker code of the trade (for Canadian, Finnish, Mexican, Philippine, and Swedish equities only)
The API

BLPAPI

tick( { Fixed Parameters } ; includeBrokerCodes = 1)

includeRpsCodes: Setting this to 1 will return transaction codes. The following values appear: -B: A customer transaction where the dealer purchases securities from the customer. -S: A customer transaction where the dealer sells securities to the customer. -D: An inter-dealer transaction (always from the sell side)

tick( { Fixed Parameters } ; includeRpsCodes = 1)

includeBicMicCodes: Setting this to 1 will return bank or market identifier code

tick( { Fixed Parameters } ; includeBicMicCodes = 1)

Multiple optional parameters can be present in a request

tick(session, ticker, eventTypes, startDateTime, endDateTime, includeConditionCodes = 1, includeExchangeCodes = 1)

Intraday Bar Data

#Fixed Parameters of Intraday Bar Data request:
#Type the ticker name you want to send request of:
ticker = "IBM US Equity"

#Type the eventType name you want to send request for:
eventType = "TRADE"

#startDateTime in the YYYY-MM-DDThh:mm:ss format
startDateTime = "2015-10-27T15:55:00"

#endDate in the YYYY-MM-DDThh:mm:ss format
endDate = "2015-10-27T16:00:00"

#Calling the function with fixed parameters:
#Then, call the tick function by providing session, tickers, # fields, startDateTime and endDate variables
bar(session, ticker, eventType, startDateTime, endDate)

The response from tick function is a Julia type BarDataResponse object. BarDataResponse contains many response elements of type TickDataElement. Each BarDataElement has following variables:

valueVar (Dependent on eventType passed)
openVar ( Float )
highVar ( Float )
lowVar ( Float )
closeVar ( Float )
volumeVar ( Float )
The API

numEventsVar ( Integer )

Default values for these variables is a string having length zero. This indicates that response don’t have data for those variables.

#initializing ticker and eventType to be passed in bar function
ticker = "IBM US Equity"
eventType = "TRADE"

#initializing startDateTime and endDateTime variables
startDateTime = "2015-10-27T15:55:00"
endDateTime = "2015-10-27T16:00:00"

#getting the response in variable ‘Response’
Response = bar(Session, ticker, eventType, startDateTime, endDateTime)

#extracting response elements from variable ‘Response’.
responseElementFirst = Response[DateTime(2015, 10, 27, 15, 55)]
responseElementSecond = Response[DateTime(2015, 10, 27, 15, 56)]
responseElementSecond = Response[DateTime(2015, 10, 27, 15, 57)]

#extracting data from response elements
firstValue = responseElementFirst.valueVar
secondValue = responseElementSecond.valueVar

NOTE: Whenever the data is not available for provided DateTime object in
the ‘Response’, ArgumentNullException() will be thrown, when extracting response
elements.

Value = Response[DateTime(1999, 10, 27, 15, 55)]
Value = Response[DateTime(2015, 10, 27, 15, 55, 5)]

(Default interval is 1, so there will not be any data at 2015-10-27T15:55:05)

All the above attempts will throw ‘ArgumentNullException()”

Optional Parameters:

returnEids: Setting this to 1 will return the entitlement identifiers (EIDs)
associated with security

bar( { Fixed Parameters } ; returnEids = 1)

interval: Sets the length of each time bar in the response. Entered as a whole
number, between 1 and 1440 in minutes. If omitted, the request will default to
one minute

bar( { Fixed Parameters } ; interval = 100)
gapFillInitialBar: When set to 1, a bar contains the previous bar values if there was no tick during this time interval

\[ \text{bar( \{ Fixed \ Parameters \} ; \ gapFillInitialBar = 1) } \]

adjustmentNormal: Setting this to 1 will adjust historical pricing to reflect: Regular Cash, Interim, 1st Interim, 2nd Interim, 3rd Interim, 4th Interim, 5th Interim, Income, Estimated, Partnership Distribution, Final, Interest on Capital, Distribution, Prorated

\[ \text{bar( \{ Fixed \ Parameters \} ; \ adjustmentNormal = 1) } \]

adjustmentAbnormal: Setting this to 1 will adjust historical pricing to reflect: Special Cash, Liquidation, Capital Gains, Long-Term Capital Gains, Short-Term Capital Gains, Memorial, Return of Capital, Rights Redemption, Miscellaneous, Return Premium, Preferred Rights Redemption, Proceeds/Rights, Proceeds/Shares, Proceeds/Warrants

\[ \text{bar( \{ Fixed \ Parameters \} ; \ adjustmentAbnormal = 1) } \]

adjustmentSplit: Setting this to 1 will adjust historical pricing and/or volume to reflect: Spin-Offs, Stock Splits/Consolidations, Stock Dividend/Bonus, Rights Offerings/Entitlement

\[ \text{bar( \{ Fixed \ Parameters \} ; \ adjustmentSplit = 1) } \]

adjustmentFollowDPDF: Setting to 1 will follow the DPDF BLOOMBERG PROFESSIONAL service function. 1 is the default setting for this option.

\[ \text{bar( \{ Fixed \ Parameters \} ; \ adjustmentFollowDPDF = 1) } \]

Multiple optional parameters can be present in a request

\[ \text{bar(} \text{session, ticker, eventType, startDateTime, endDateTime, interval = 100, adjustmentSplit = 1) } \]

BLOOMBERG is a registered trademark of Bloomberg Finance L.P. or its affiliates.