OPEN MARKET DATA INITIATIVE

BLPAPI Use Case: Subscription Data

February 1, 2012

Bloomberg

CONTENTS

02 BLPAPI PROGRAMMING INTERFACE

02 SUBSCRIPTION DATA OVERVIEW

03 TOPICS

04 C++ EXAMPLE: WEIGHTED EQUITY INDEX

BLPAPI Use Case: Subscription Data

BLPAPI PROGRAMMING INTERFACE

BLPAPI is the programming interface for Bloomberg's Desktop and Server API products as well as for Managed B-Pipe and Platform (http://open.bloomberg.com). Built on a flexible service-oriented architecture, BLPAPI supports both request/response and publish/subscribe paradigms. Subscription services are used to receive data that changes frequently and/or at unpredictable intervals, for example equity pricing data or order-fill notifications. This paper shows how to use subscription data.

SUBSCRIPTION DATA OVERVIEW

The following diagram shows how subscriptions fit into BLPAPI. The concept of a "topic" is central. Topics identify data to be delivered through a subscription (security and fields) and provide applications a way to specify options controlling how to receive the data. The interface provides the mechanism for services to implement features such as intervalization, a form of data conflation where periodic updates are delivered. As will be explained below, the separate "topic resolution" step also facilitates data sharing across applications and users, further reducing bandwidth requirements.

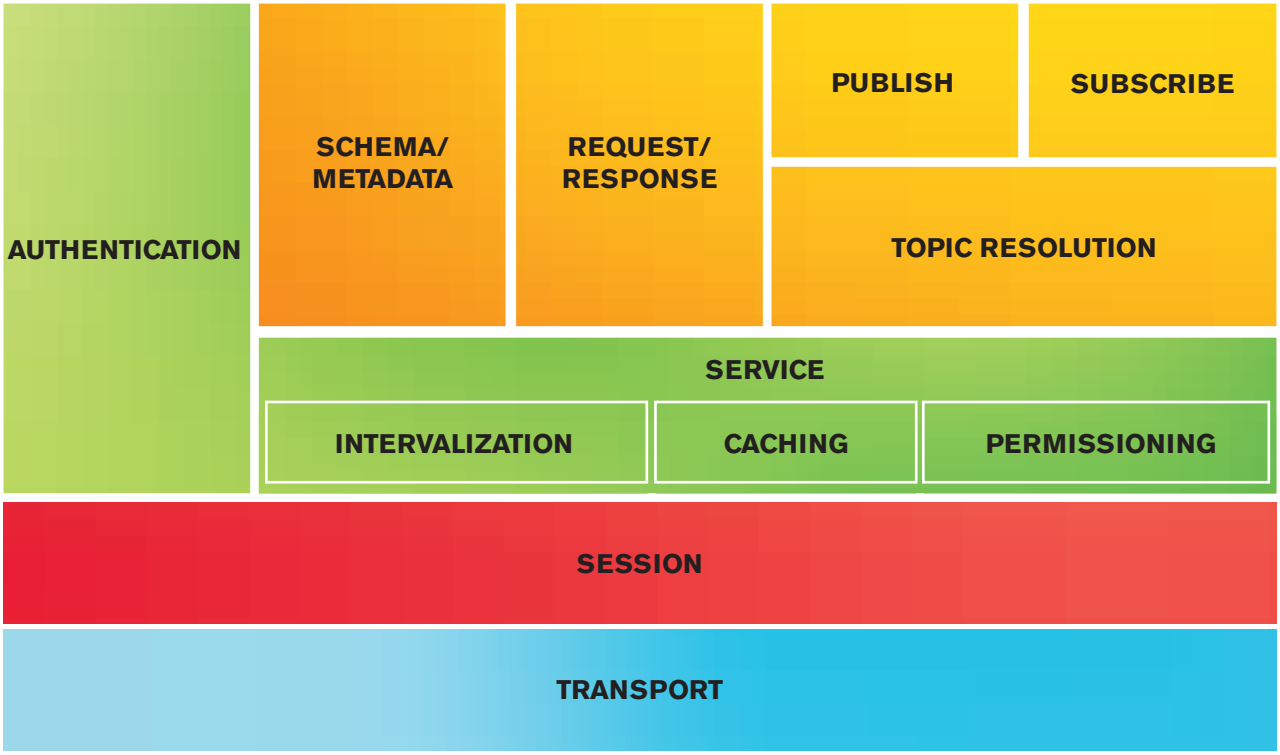

Figure 1. BLPAPI Interface Components

Subscriptions are used to convey asynchronous data, where the timing of the generation of data is independent of its consumption by client applications. Although internally event-driven and multi-threaded, BLPAPI supports both synchronous and asynchronous programming models.

BLPAPI Use Case: Subscription Data

Prior to using subscription or other services, an application must create a session; in some deployments, this may require a separate authentication step. The following sequence diagram shows the session establishment process:

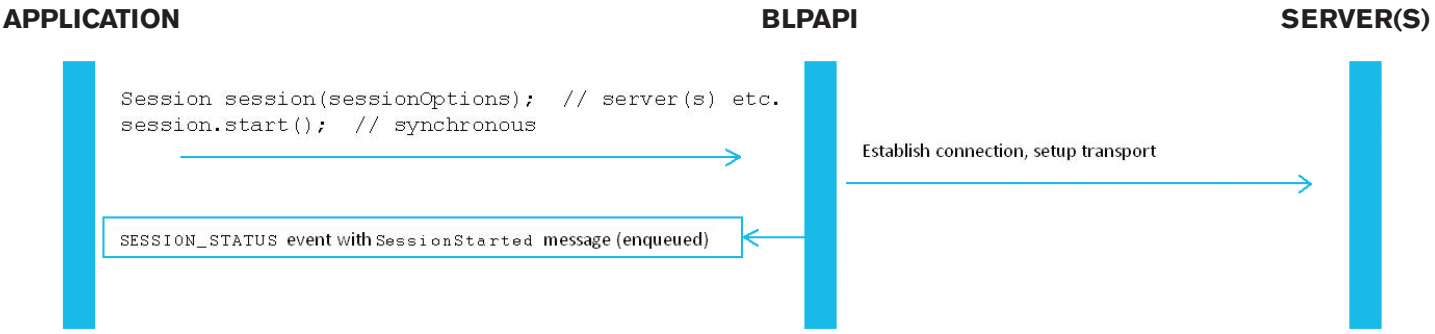

Figure 2. Session establishment (using synchronous API)

TOPICS

Topic strings are composed of three parts:

- **Service Name –** Identifying services in this way provides applications with location-agnostic routing. For example, Bloomberg's market data feed is accessed under the "//blp/mktdata" service.
- **Security Symbol –** Instruments are specified using an (optional) symbology identifier, followed by a security identifier. For example, "/cusip/459200101" and "/bsid/399432473346" identify IBM common stock. (The latter, Bloomberg's open symbology, is open and free for unrestricted use; BSYM is available publically at http://bsym.bloomberg.com.) Importantly, the two topics specified as above resolve to the same (internal) identifier, allowing the data to be sent once to each site, host, or application.
- **Subscription Options –** These affect the content that is delivered; their support is service-dependent. Bloomberg's services are built on a common infrastructure and generally support field-based subscriptions and intervalization options, for example:
  - "fields=LAST_PRICE,SIZE_LAST_TRADE" means that updates not affecting the specified fields can be filtered out (in particular, bid/ask quotes, which comprise the bulk of market data ticks).
  - "interval=2.0" specifies that data is conflated to at least two seconds between updates.

The following Java code shows how to establish a subscription:

```
SubscriptionList subscriptions = new SubscriptionList();
subscriptions.add(new Subscription(
    "//blp/mktdata/bsid/399432473346",  // IBM US Equity
    "LAST_PRICE,SIZE_LAST_TRADE",       // fields
    "interval=2.0",                     // other options
    new CorrelationID(40)));  // application's security identifier
session.subscribe(subscriptions);
```

Additional interfaces allow different combinations of topic constituents to be conveniently specified. They can also be provided as a single concatenated string.

BLPAPI Use Case: Subscription Data

C++ EXAMPLE: WEIGHTED EQUITY INDEX

The following C++ example subscribes to a collection of stocks and computes, on each update, a weighted index representing their aggregate price.

>> Code Listing

The application conveniently selects security "correlation ids" that map to entries in a table of index constituents:

```
#include <blpapi_defs.h>
#include <blpapi_event.h>
#include <blpapi_message.h>
#include <blpapi_correlationid.h>
#include <blpapi_session.h>
#include <blpapi_subscriptionlist.h>

#include <iostream>

using namespace BloombergLP::blpapi;

int main(int argc, char **argv)
{
    bool debugTrace = (argc > 1);  // if any arguments specified, output trace

    // construct a weighted index of a few beverage stocks
    struct IndexMember {
        const char *d_security;
        double      d_weight;
        double      d_price;
    } indexMembers[] = {
        { "KO US Equity",  2.0, 0.0 },  // Coca-Cola
        { "PEP US Equity", 1.0, 0.0 },  // Pepsi
        { "DPS US Equity", 0.5, 0.0 }   // Dr. Pepper/Snapple
    };

    SessionOptions sessionOptions;  // use defaults for most settings
    sessionOptions.setDefaultSubscriptionService("//blp/mktdata");

    Session session(sessionOptions);
    if (!session.start()) {
        std::cerr << "Failed to start session." << std::endl;
        return -1;
    }

    if (!session.openService("//blp/mktdata")) {
        std::cerr << "Failed to open //blp/mktdata." << std::endl;
        return -1;
    }

    // subscribe to LAST_PRICE field for all index members
    SubscriptionList subscriptions;
    for (int i = 0; i < sizeof(indexMembers)/sizeof(indexMembers[0]); ++i) {
        // correlation id is offset in 'indexMembers' array
        subscriptions.add(indexMembers[i].d_security, "LAST_PRICE", "", CorrelationId(i));
    }

    session.subscribe(subscriptions);  // submit the subscription request

    double index = 0.0;  // weighted index value (valid after all initial paints received)

    // optimization: create a handle for the LAST_PRICE field
    Name LAST_PRICE("LAST_PRICE");
```

BLPAPI Use Case: Subscription Data

```
    while (true) {
        Event event = session.nextEvent();
        MessageIterator msgIter(event);
        while (msgIter.next()) {
            Message msg = msgIter.message();
            if (debugTrace) {
                msg.print(std::cout) << std::endl;
            }

            if (Event::SUBSCRIPTION_DATA == event.eventType() &&
                msg.hasElement(LAST_PRICE, true)) {
                // find application information corresponding to this security
                IndexMember &member = indexMembers[msg.correlationId().asInteger()];

                double price = msg.getElementAsFloat64(LAST_PRICE);  // extract price

                // recalculate the index and output value
                index += (price - member.d_price) * member.d_weight;
                member.d_price = price;
                std::cout << "Index = " << index << std::endl;
            }
        }
    }

    return 0;
}
```

Sample Output

Sample output follows:

```
Index = 65.44
Index = 200.3
Index = 219.655
Index = 219.636
Index = 219.636
Index = 219.636
Index = 219.635
...
```

If the application is run with full message-level trace, then we can see the events that come in during session/service and subscription setup:

```
SessionConnectionUp = {
    server = 127.0.0.1:8194
}

SessionStarted = {
}

ServiceOpened = {
    serviceName = //blp/mktdata
}

SubscriptionStarted = {
    exceptions[] =

}

SubscriptionStarted = {
    exceptions[] =

}

SubscriptionStarted = {
    exceptions[] =

}
```

BLPAPI Use Case: Subscription Data

An initial paint, categorized as a “summary” event, will arrive for each subscription. As shown below, this may contain data for fields not explicitly subscribed. (If a field is not subscribed, the application is not guaranteed to get current values.)

```
MarketDataEvents = {
    MKTDATA_EVENT_TYPE = SUMMARY
    MKTDATA_EVENT_SUBTYPE = INITPAINT
    PX_METHOD = PRC
    BID = 67.420000
    ASK = 67.430000
    BEST_BID = 67.420000
    BEST_ASK = 67.430000
    IND_BID_FLAG = false
    IND_ASK_FLAG = false
    ASK_SIZE_TDY = 9
    BID_SIZE_TDY = 20
    BID_TDY = 67.420000
    PRICE_LAST_BID_RT = 67.420000
    ASK_TDY = 67.430000
    PRICE_LAST_ASK_RT = 67.430000
    ASK_SIZE = 9
    BID_SIZE = 20
    EXCH_CODE_BID = O
    BID_PX_LOCAL_EXCH_SOURCE_RT = UO
    PRICE_PREVIOUS_CLOSE_RT = 67.460000
    ID_BB_SEC_NUM_SRC = 399432473547
    LAST_PRICE = 67.430000
    LAST_TRADE = 67.430000
    VOLUME = 3649438
    HIGH = 67.980000
    LOW = 67.180000
    OPEN = 67.980000
    OPEN_TDY = 67.980000
...
```

Subsequent updates will be much smaller. In the case of intervalized subscriptions, summary updates, with subtype “interval,” will only contain field values that have changed. Otherwise, updates will contain fields related to the market event, e.g. a trade:

```
MarketDataEvents = {
    MKTDATA_EVENT_TYPE = TRADE
    MKTDATA_EVENT_SUBTYPE = NEW
    EQY_TURNOVER_REALTIME = 246192516.310000
    TURNOVER_TODAY_REALTIME = 246192516.310000
    LAST2_TRADE = 67.420400
    LAST_PRICE = 67.420700
    LAST_ALL_SESSIONS = 67.420700
    LAST_TRADE_TDY = 67.420700
    LAST2_DIR = -1
    PRICE_LAST_RT = 67.420700
    PRICE_LAST_TRADE_RT = 67.420700
    LAST_DIR = 1
    LAST_TRADE = 67.420700
    SIZE_LAST_TRADE = 100
    SIZE_LAST_TRADE_TDY = 100
    TRADE_SIZE_ALL_SESSIONS_RT = 100
    ALL_PRICE_SIZE = 100
    ALL_PRICE_COND_CODE =
    ALL_PRICE = 67.420700
    LAST_PRICE_COND_CODE_RT =
    VOLUME = 3650038
...
```

To save bandwidth, fields can be calculated and generated locally rather than being sent over the network from the service provider to the client. For example, the VOLUME field above could be accumulated and emitted from a system at the client’s site.

To learn more about the Bloomberg Open Market Data Initiative, visit open.bloomberg.com. Questions and comments about BLPAPI can be sent to open-tech@bloomberg.net. Questions and comments about BSYM can be sent to bsym@bloomberg.net.

open.bloomberg.com

New York	**London**	**Frankfurt**	**San Francisco**
+1 212 318 2000	**+44 20 7330 7500**	**+ 49 69 9204 1210**	**+1 415 912 2960**
Hong Kong	**Sao Paulo**	**Singapore**	**Tokyo**
+852 2977 6000	**+55 11 3048 4500**	**+65 6212 1000**	**+81 3 3201 8900**