FACTSET

FactSet Tick History OnDemand Service

Programmer's Manual and Reference
Version 1.0 k

Table of Contents

Notice

This manual contains confidential information of FactSet Research Systems Inc. or its affiliates ("FactSet"). All proprietary rights, including intellectual property rights, in the Licensed Materials will remain property of FactSet or its Suppliers, as applicable. The information in this document is subject to change without notice and does not represent a commitment on the part of FactSet. FactSet assumes no responsibility for any errors that may appear in this document.

FactSet Consulting Services

North America - FactSet Research Systems Inc.	
United States and Canada	+1.877.FACTSET
Europe – FactSet Limited	
United Kingdom	0800.169.5954
Belgium	800.94108
France	0800.484.414
Germany	0800.200.0320
Ireland, Republic of	1800.409.937
Italy	800.510.858
Netherlands	0800.228.8024
Norway	800.30365
Spain	900.811.921
Sweden	0200.110.263
Switzerland	0800.881.720
European and Middle Eastern countries not listed above	+44.(0)20.7374.4445
Pacific Rim- FactSet Pacific Inc.	
Japan Consulting Services (Japan and Korea)	0120.779.465 (Within Japan) +81.3.6268.5200 (Outside Japan)
Hong Kong Consulting (Hong Kong, China, India, Malaysia, Singapore, Sri Lanka, and Taiwan)	+852.2251.1833
Sydney Consulting Services	1800.33.28.33 (Within Australia) +61.2.8223.0400 (Outside Australia)
E-mail Support	
	support@factset.com

Document Organization and Audience

This document describes how to use the FactSet Exchange DataFeed Snapshot History Service. You should be familiar with the XML language, the HTTP protocol, and Web Services. This document will describe the syntax needed for proper request formatting as well as the rules for processing responses. In addition, complete code examples are included, which further illustrate the use of this service.

- Chapter 1 is an introduction to the Exchange Snapshot History Service with general concepts and terminology.
- Chapter 2 explains how to request data using the HTTP GET method.
- Chapter 3 details the XML and CSV responses, as well as the types of errors that can be received.
- Chapter 4 describes how to use this service with FactSet Data Link
- The Appendix supplements this document by providing additional code examples and a list of recent changes to the document.

Document Conventions

This document uses the following conventions:

- Code examples use a courier 10 font - `int i = 0;`
- Items of importance will be in boxes of following type:

> ❖ ***Important notations will be in this type of box.***

Trademarks

Acknowledgements

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org).

1 Introduction to General Concepts

1.1 Introduction

The Exchange DataFeed Snapshot History service is designed to provide a dynamic way to access real-time trading details for a specific security. This data comes from FactSet's Time and Sales database, which provides history of all quotes and trades for a trailing 6 months[1] period[2]. The Snapshot History service is designed to expose this data to various tools outside of the FactSet workstation. Using DataDirect technology, this dataset can be easily accessed in any environment that can support an https call.

The data provided from this service, often called 'tick' data, is useful in analyzing the volume and price transactions in a given trading period. This analysis can provide insight into how to optimize transaction costs.

[1] Due to the limited history the links in this document may not always be up to date. Please make sure to adjust the dates in the links to use a date within the 6 months.

[2] Full trade and quote data for options is available for 30 days. Trades only data (no quotes) is available for a full year for Equity

1.2 FactSet Tick History OnDemand Service

The FactSet Tick History OnDemand Service provides synchronous access to data via the standard HTTPS protocol (see *Figure 1*). Data is returned as a well-described XML document or CSV file. Responses for multiple data elements can be forced into a table-like structure, which further simplifies application development.

This service uses the FactSet DataFeed API and thus the same data model. Chapter 5 describes the data fields, types, and possible values for each type of record.

Clients of the Exchange Snapshot History Service will be given a secure URL, username, and password. The user information is used in each request to authenticate and permission each data item.

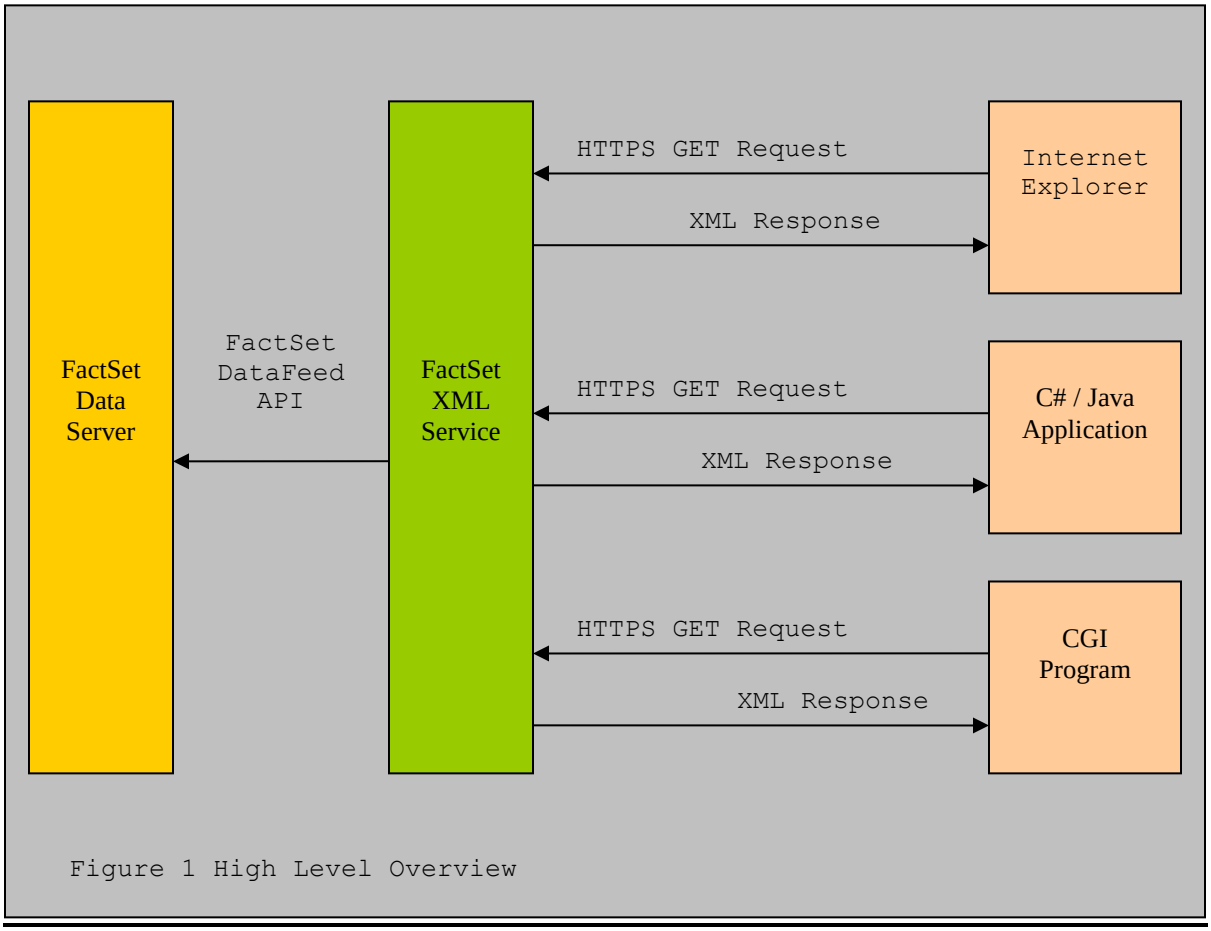

Figure 1 High Level Overview

1.3 Terminology

The following terminology is used throughout this documentation:

Terminology	Meaning
API	Application Programming Interface - a set of defined interfaces that applications use to extract information from the FactSet Data Server.
SDK	Software Development Kit - a collection of libraries, include files, documentation, and sample code that makes up this toolkit.
XML	eXtensible Markup Language - a defined standard for exchanging information. The information contains markup tags used to describe the data values.
CSV	Comma Separated Value - a table-like structure that separates fields via a delimiter (usually a comma ","). The first header row describes each of the column names.
TCP/IP	Transport Control Protocol over Internet Protocol - one of the protocols that this service uses to communicate with the FactSet Data Server.
FactSet Data Server	A server that provides permissioned access to FactSet data.
FDS	Multiple meanings. FDS is the ticker symbol for FactSet Research Systems Inc. It is also the C++ namespace that encapsulates the FactSet DataFeed API. Finally, it may stand for the FactSet Data Server. The meaning is defined by its context.
Service	A data source or supplier identified by a string name.
Opaque Data	Data without a defined interpretation, which is simply a pointer to data and a size.
Field/Value Pairs	A self-describing message format used in API responses. Each pair contains a FID and some opaque data. The FID defines the type and meaning of the data.
HTTP	Hypertext Transfer Protocol - the protocol that governs how information is exchanged over a network.
HTTPS	Secure Hypertext Transfer Protocol - the protocol that uses SSL to securely communicate HTTP messages.
SSL	Secure Sockets Layer - a protocol used to encrypt data over an insecure medium.
URL	Uniform Resource Locator - used to identify a resource when using HTTP on the Internet.

1.4 Exchange Snapshot History Service Core Functionality and Benefits

The Exchange Snapshot Service provides the following features to applications:

- Synchronous access using standard HTTP methods.
- Consistent data model that matches the underlying C++ DataFeed API.
- Stateless requests (No sticky cookies).
- Platform independence.
- Zero-based installation.
- Batched requests allowing multiple items to be requested at the same time.
- Permissioned data access.
- Seamless failover.
- Production and Beta environments.
- Easy access to FactSet's web-based products through http requests.

1.4.1 Synchronous Access

Synchronous data access simplifies application programming and makes it straightforward to exploit the HTTP protocol.

1.4.2 Stateless Request/Response

The Tick History Service is a stateless request/response service. The service does not make any use of HTTP cookies, and thus allows requests to be load-balanced over many servers. This in turn, improves both throughput and response time.

1.4.3 Platform Independence

The Tick History Service can be used without the any FactSet-supplied software. Applications can access data from any system that supports TCP/IP.

1.4.4 Zero-Based Installation

Proprietary software is not needed to access this service. Any standard library or application that can issue HTTP queries can request data via the Tick History Service.

1.4.5 Permissioned Access

The Exchange Snapshot History Service handles all exchange permissioning on behalf of the user defined in each request. Therefore, multi-user proxy-type applications can easily be created, provided that the users are already using FactSet applications.

1.4.6 Seamless Failover

If a FactSet Data Server or an entire data center goes down, requests will be routed to an available Data Server using state-of-the-art load balancers. You do not have to change your software, as this failover will be automatic.

1.4.7 Production and Beta Environments

FactSet provides two independent environments for programmatic access. One is for production and the other for Beta. The following table shows the URLs needed for each system.

Type	Host Name	Service	Example URL[3]
Production	datadirect.factset.com	TickHistory	https://datadirect.factset.com/services/TickHistory
Beta	datadirect-beta.factset.com	TickHistory	https://datadirect-beta.factset.com/services/TickHistory

- **You will start on the beta environment until your software is ready for production. Therefore, this document will use the beta URL for all examples and sample codes. Once you move to a production environment, you will need to update both the URL and data service.**
- **You should contact FactSet to ensure your production permissions meet your requirements as permissions in the beta environment may differ from the production environment.**

1.4.8 Easy Access through DataDirect

Since the TickHistory Service is part of FactSet OnDemand product suite, it provides easy access to FactSet data through web requests. All services are authenticated by a central source, so the same username and password will work for all services. In addition, naming conventions have been standardized to simplify the consumption of multiple services.

Basic HTTP Authentication is used. This scheme requires the following HTTP header be added to the request:

Authorization: Basic {Base-64 Encoding of "user:password"}

The exact header will be provided by FactSet along with the FactSet OnDemand username and password.

- **While the standard scheme is HTTP Basic Authentication, there are stronger types of authentication provided by FactSet if preferred.**

[3] The example URLs demonstrate the different services assigned to production and beta. The URLs given above may not be valid. For example, the query string is missing in the above example.

2 HTTP GET Requests[4]

2.1 HTTP GET

When using an HTTP GET request, the search criteria is sent via the query string in the URL.

Example
This request returns all available fields for the symbol IBM-USA on 14/10/2017[5] between 10:00 AM and 3:00 PM in one hour intervals.

https://datadirect-beta.factset.com/services/TickHistory?id=IBM-USA&date=20170914&st=100000&et=150000&=&interval=1H&format=xml

This request returns the LAST_1, LAST_TIME_1, LAST_VOL_1 fields for the symbols IBM-USA on 14/10/2017 between 10:00 AM and 3:00 PM in one hour intervals.

https://datadirect-beta.factset.com/services/TickHistory?id=IBM-USA&date=20170914&st=100000&et=140000&=&interval=1H&format=xml&fields=last_1,last_time_1,last_vol_1

2.2 Query Variables

Query Variable	Description	Default Value	Allowable Values
Req_Id	Request Identification String. Can be used by the application to keep track of requests. The id is not used by this service, however, it is included in the XML response.	None	Any integer value
Format	The format of the output file.	XML	XML – XML Response CSV – CSV Response JSON – JSON Response
id	Requested symbol or security. The symbol can be a FactSet exchange symbol, CUSIP, or SEDOL. **NOTE:** Only one identifier can be requested per request	None	A single identifier
Fields	Requested fields. This is also a comma-separated list (no spaces).	All Fields	See Section 2.2.1 Below
Sd/Date	This is the start date for the Snapshot History. Requests should be made in the format YYYYMMDD.	Current Business Day	Any acceptable specific date
Ed	This is the end date for the Snapshot History. Requests should be made in the format YYYYMMDD. If the period requested is larger than Max. Days requested, results will be limited to the Max Days starting at sd/date	Sd/date	Any acceptable specific date
St	This is the start time for the Snapshot History requested. Requests should be made in a HHMMSS format.	040000	Possible values range from '000000' – '235959'.
Et	Requested end Time. This is the end time for the Snapshot History requested. Requests should be made in a HHMMSS format.	200000	Possible values range from '000000' – '235959'.
Interval	Requested interval. This is the interval the data returned from the service. Possible values range from 1 second to 1H.	1 minute	See Section 2.2.2 Below
include_request	Flag to include request id and key in the CSV output	False	True/T, False/F
dataset	Decides what type of trades to return, default is all trades and quotes, alternatively charting can be selected which will filter out unofficial trades.	<blank>	<blank>/charting

[4] HTTP POST requests are not supported for the TickHistory service.
[5] Please make sure that all requests contain a date within the past year. The Snapshot History service only provides 1 year of history, so requests for earlier dates will not work correctly.

2.2.1 Allowed Values for fields

Field	Description
TIME	The exchange-local time of the trade or quote, or the time of an interval
BID_1	The last bid price or last bid price in an interval
BID_VOL_1	The volume of the last bid or last bid in an interval
BID_EXCH_1	The exchange of the last bid or last bid in an interval
ASK_1	The last ask price or last ask price in an interval
ASK_VOL_1	The volume of the last ask or last ask in an interval
ASK_EXCH_1	The exchange of the last ask or last ask in an interval
LAST_1	The last trade price or last trade price in an interval
LAST_DATE_1	The date of the last trade or last trade in an interval
LAST_TIME_1	The time of the last trade or last trade in an interval
LAST_VOL_1	The trade volume, or the sum of all trade volumes inside a bin.
LAST_EXCH_1	The exchange of the last trade or last trade in an interval
CUM_VOL	The symbol's daily cumulative volume, or the last cumulative volume in an interval
INT_CVOL	Interval cumulative volume, i.e. the sum of all volumes in an interval
VWAP	The daily volume weighted average price, or the last VWAP in an interval
INT_VWAP	Interval VWAP, i.e. the VWAP of the trades in an interval.
OPEN_1	The first trade of an interval
HIGH_1	The highest trade price in an interval
LOW_1	The lowest trade price in an interval

2.2.2 Allowed values for intervals

Interval	Description	Max. Days Requested[6]
0	Every tick	1
1S	1 Second	1
5S	5 Seconds	1
10S	10 Seconds	1
15S	15 Seconds	1
30S	30 Seconds	15
1M	1 Minute	30
2M	2 Minutes	60
5M	5 Minutes	60
10M	10 Minutes	60
15M	15 Minutes	60
30M	30 Minutes	60
1H	1 Hour	60

- **If the number of days requested exceeds the "Max Days Requested" limit listed above, only data for the days requested first in the date range will be returned.**
- **Only one ticker may be specified in each request.**

[6] The Maximum Days Requested is the total number of days that be requested at any given interval. If more than the specified days is requested the results will be limited to Max. days starting at sd/date

- **A maximum of 10 Tick History requests and a maximum of 100,000 rows of data may be made in any one-minute period[7].**

2.3 Example Responses

2.3.1 XML Response with Specified Fields (format = 'xml')

This request returns the LAST_1, LAST_TIME_1, LAST_VOL_1 fields for the symbols IBM-USA on 14/09/2017 between 10:00 AM and 3:00 PM in one hour intervals:

https://datadirect-beta.factset.com/services/TickHistory?id=IBM-USA&date=20170914&st=100000&et=140000&=&interval=1H&format=xml&fields=last_1,last_time_1,last_vol_1

```
<Response>
    <Request>
        <RequestedSymbol>IBM-USA</RequestedSymbol>
        <RequestedFields>last_1,last_time_1,last_vol_1</RequestedFields>
        <RequestKey>4EDFD64341D51524</RequestKey>
    </Request>
    <Error code="0" description="" />
    <Records key="IBM-USA">
        <Record>
            <Fields>
                <Field id="300" name="LAST_1" value="185.51" />
                <Field id="302" name="LAST_TIME_1" value="100000" />
                <Field id="304" name="LAST_VOL_1" value="1013024" />
            </Fields>
        </Record>
        <Record>
            <Fields>
                <Field id="300" name="LAST_1" value="185.83" />
                <Field id="302" name="LAST_TIME_1" value="110000" />
                <Field id="304" name="LAST_VOL_1" value="418542" />
            </Fields>
        </Record>
        <Record>
            <Fields>
                <Field id="300" name="LAST_1" value="185.56" />
                <Field id="302" name="LAST_TIME_1" value="120000" />
                <Field id="304" name="LAST_VOL_1" value="419045" />
            </Fields>
        </Record>
        <Record>
            <Fields>
                <Field id="300" name="LAST_1" value="185.72" />
                <Field id="302" name="LAST_TIME_1" value="130000" />
                <Field id="304" name="LAST_VOL_1" value="501591" />
            </Fields>
        </Record>
```

[7] An additional subscription is available to be able to access 20 requests/200 000 rows of data per minute. Contact your FactSet representative for further information.

```
    </Records>
</Response>
```

2.3.2 CSV Response with Specified Fields (format = 'csv')

This request is for the same data as the previous example, with the exception of the response format parameter.

https://datadirect-beta.factset.com/services/TickHistory?id=IBM-USA&date=20170914&st=100000&et=140000&=&interval=1H&format=csv&fields=last_1,last_time_1,last_vol_1

The URL above produces the following CSV response:

```
LAST_1,LAST_TIME_1,LAST_VOL_1
185.51,100000,1013024
185.83,110000,418542
185.56,120000,419045
185.72,130000,501591
```

2.3.3 XML Response with All Fields (format = 'xml')

This example contains all the fields for the symbol FDS-USA (since the fields variable is not present).

https://datadirect-beta.factset.com/services/TickHistory?id=IBM-USA&date=20170914&st=100000&et=100500&=&interval=5M&format=xml


```
<Response> -<Request id="">
                <RequestedSymbol>IBM-USA</RequestedSymbol>

        <RequestedFields>BID_1,BID_VOL_1,BID_EXCH_1,ASK_1,ASK_VOL_1,ASK_EXCH_1,LAST_1,LAST_DATE_1,LAST_TIME_
1,LAST_VOL_1,LAST_EXCH_1,CUM_VOL,VWAP,OPEN_1,HIGH_1,LOW_1,TRADE_CONDITION,GMT_OFFSET,PRICE_CURRENCY</Re
questedFields>
                <RequestKey>541042DE6C0F6E82</RequestKey>
        </Request>
        <Error description="" code="0"/> -<Records stale="" key="IBM-USA" req_sym="IBM-USA"> -<Record> -<Fields>
                                <Field id="100" value="190.27" name="BID_1"/>
                                <Field id="104" value="1" name="BID_VOL_1"/>
                                <Field id="107" value="11106" name="BID_EXCH_1"/>
                                <Field id="200" value="190.31" name="ASK_1"/>
                                <Field id="204" value="1" name="ASK_VOL_1"/>
                                <Field id="207" value="10050" name="ASK_EXCH_1"/>
                                <Field id="300" value="190.27" name="LAST_1"/>
                                <Field id="301" value="20170909" name="LAST_DATE_1"/>
                                <Field id="302" value="100453990" name="LAST_TIME_1"/>
                                <Field id="304" value="20428" name="LAST_VOL_1"/>
                                <Field id="307" value="11009" name="LAST_EXCH_1"/>
                                <Field id="601" value="355797" name="CUM_VOL"/>
                                <Field id="603" value="190.17" name="VWAP"/>
                                <Field id="710" value="190.02" name="OPEN_1"/>
                                <Field id="720" value="190.3" name="HIGH_1"/>
                                <Field id="723" value="189.98" name="LOW_1"/>
                                <Field id="2709" value="01 2 24" name="TRADE_CONDITION"/>
                                <Field id="2037" value="-240" name="GMT_OFFSET"/>
                                <Field id="2032" value="USD" name="PRICE_CURRENCY"/>
                        </Fields>
                </Record> -<Record> -<Fields>
                                <Field id="100" value="190.28" name="BID_1"/>
                                <Field id="104" value="1" name="BID_VOL_1"/>
                                <Field id="107" value="10050" name="BID_EXCH_1"/>
                                <Field id="200" value="190.35" name="ASK_1"/>
                                <Field id="204" value="1" name="ASK_VOL_1"/>
                                <Field id="207" value="11009" name="ASK_EXCH_1"/>
                                <Field id="300" value="190.28" name="LAST_1"/>
                                <Field id="301" value="20170909" name="LAST_DATE_1"/>
                                <Field id="302" value="100558948" name="LAST_TIME_1"/>
                                <Field id="304" value="2500" name="LAST_VOL_1"/>
                                <Field id="307" value="10050" name="LAST_EXCH_1"/>
                                <Field id="601" value="359361" name="CUM_VOL"/>
                                <Field id="603" value="190.171" name="VWAP"/>
                                <Field id="710" value="190.26" name="OPEN_1"/>
                                <Field id="720" value="190.3" name="HIGH_1"/>
                                <Field id="723" value="190.225" name="LOW_1"/>
                                <Field id="2709" value="01 2 24" name="TRADE_CONDITION"/>
                                <Field id="2037" value="-240" name="GMT_OFFSET"/>
                                <Field id="2032" value="USD" name="PRICE_CURRENCY"/>
                        </Fields>
                </Record>
```

```
        </Records>
</Response>
```

2.3.4 CSV Response with All Fields (format = 'csv')

This example contains all the fields for the symbol FDS-USA in a CSV response (since the fields variable is not present).

https://datadirect-beta.factset.com/services/TickHistory?id=IBM-USA&date=20170914&st=100000&et=100500&=&interval=5M&format=csv

The URL above produces the following CSV response:

```
BID_1,BID_VOL_1,BID_EXCH_1,ASK_1,ASK_VOL_1,ASK_EXCH_1,LAST_1,LAST_DATE_1,LAST_TIME_1,LAST_VOL_1,LAST
_EXCH_1,CUM_VOL,VWAP,OPEN_1,HIGH_1,LOW_1,TRADE_CONDITION,GMT_OFFSET,PRICE_CURRENCY
190.27,1,11106,190.31,1,10050,190.27,20170909,100453990,20428,11009,355797,190.17,190.02,190.3,189.98,01 2
24,-240,USD
190.28,1,10050,190.35,1,11009,190.28,20170909,100558948,2500,10050,359361,190.171,190.26,190.3,190.225,01 2
24,-240,USD
```

2.4 Example Codes for HTTP GET

All of the example codes request the LAST_1, LAST_TIME_1, and LAST_VOL_1 fields for the symbol FDS-USA at 60 minute intervals. Two programming environments are illustrated (C#/.NET and Java). Each programming language has three sections. The first section makes the HTTP GET request, the second parses the XML response, and the third displays the run-time output. The examples have been tested and can be used as-is provided that the HTTP Basic Authorization header is modified appropriately.[8]

2.4.1 Using C# (.NET 2.0)

Issuing an HTTP GET request

```
using System.IO;
using System.IO.Compression;
using System.Net;
using System.Xml;

class MyXMLParser
{

[System.STAThread]
static void Main(string [] args)
{
  // setup the request
  HttpWebRequest req = (HttpWebRequest)WebRequest.Create(
      "https://datadirect-beta.factset.com/services/TickHistory?" +
      "id=FDS-USA&format=xml" +
      "&date=20170914&fields=LAST_1,LAST_TIME_1,LAST_VOL_1" +
      "&interval=1H&st=100000&et=130000"
  req.KeepAlive = false ;
  req.Headers.Add("Accept-Encoding", "deflate, gzip");
  req.Headers.Add("Authorization", "Basic AaBbCcDdEeFfGgHhIi1234==");

  // make request and get the response
  HttpWebResponse rsp = (HttpWebResponse)req.GetResponse();
  Stream stream = rsp.GetResponseStream();

  // uncompress the response
  If (rsp.ContentEncoding.Equals("deflate"))
    stream = new DeflateStream(stream, CompressionMode.Decompress);
  else if (rsp.ContentEncoding.Equals("gzip"))
    stream =  new GZipStream(stream, CompressionMode.Decompress);
  .
  .
  .
```

[8] The username and password, as well the exact HTTP header to add to the request, will be supplied by FactSet.

Parsing the XML response

This code uses a C# XMLTextReader to move through the response nodes and print out the values[9]

```
.
.
.
XmlTextReader reader = new XmlTextReader(stream);
reader.WhitespaceHandling = WhitespaceHandling.None;

  while (reader.Read())
  {
    if (reader.NodeType.Equals(XmlNodeType.Element))
    {
      if (reader.Name.Equals("Error"))
      {
        System.Console.WriteLine(reader.Name);
        printAttributes(reader);
        System.Console.WriteLine();
      }
      else if (reader.Name.Equals("Records"))
      {
        System.Console.WriteLine(reader.Name);
        printAttributes(reader);
      }
      else if (reader.Name.Equals("Record"))
      {
        System.Console.WriteLine(reader.Name);

        // Move to first Field element
        while (reader.Read() && !reader.Name.Equals("Field"))
        { }

        // Print out all field ids, names and values
        do
        {
          printAttributes(reader);
        }  while (reader.Read() && reader.Name.Equals("Field"));
        System.Console.WriteLine();
      }
    }
  }
}

static void printAttributes(XmlTextReader reader)
{
while (reader.MoveToNextAttribute())
  System.Console.Write(reader.Name + ": " + reader.Value + ", ");
System.Console.WriteLine();
}

} // class MyXmlParser
```

[9] Applications may also use an XMLDocument to parse the response, see Appendix A.

Program output

```
Error
code: 0, description: ,

Records
req_sym: FDS-USA, key:FDS-USA, stale: ,

Record
id: 300, name: LAST_1, value: 92.29,
id: 302, name: LAST_TIME_1, value: 100000,
id: 304, name: LAST_VOL_1, value: 21725,

Record
id: 300, name: LAST_1, value: 92.3,
id: 300, name: LAST_TIME_1, value: 110000,
id: 300, name: LAST_VOL_1, value: 42516,

Record
id: 300, name: LAST_1, value: 92.81,
id: 300, name: LAST_TIME_1, value: 120000,
id: 300, name: LAST_VOL_1, value: 23419,
```

2.4.2 Using Java

Issuing an HTTP GET request

This example uses Java to make a web request and parse the response.

> ❖ **Since there is a java.net.ContentHandler and a org.xml.sax.ContentHandler, applications should avoid the following code:**
> **import java.net.*;**
> **import org.xml.sax.*;**

```
import java.io.*;
import java.util.zip.*;
import java.net.URL;
import javax.net.ssl.HttpsURLConnection;
import javax.xml.parsers.*;
import org.xml.sax.*;

public class MyXmlParser implements ContentHandler {

public static void main(String[] args)
{
  try {
    // setup the request
    URL url =  new URL(
      "https://datadirect-beta.factset.com/services/TickHistory?" +
      "id=FDS-USA&format=xml" +
      "&date=20170914&fields=LAST_1,LAST_TIME_1,LAST_VOL_1" +
      "&interval=1H&st=100000&et=130000"
    HttpsURLConnection conn = HttpsURLConnection)url.openConnection();
    conn.setDoOutput(true);
    conn.setRequestProperty("Accept-Encoding", "deflate, gzip");
    conn.setRequestProperty("Authorization",
                            "Basic AaBbCcDdEeFfGgHhIi1234==");

    // getInputStream will implicitly connect and get the response
    InputStream inputStream = conn.getInputStream();

    // uncompress the response
    if (conn.getContentEncoding().equals("deflate"))
      inputStream =  new InflaterInputStream(inputStream,
                                             new Inflater(true));
    else if (conn.getContentEncoding().equals("gzip"))
      inputStream =  new GZIPInputStream(inputStream);
.
.
.
```

Parsing the XML response

The SAX parser, which comes with the Standard Edition of Java, uses event-handler callbacks for each element in the XML document[10]. Therefore, in order to print the record attributes for each field, the "req_sym", "key", and "stale" values must be saved in temporary variables.

```
    .
    .
    XMLReader reader =
        SAXParserFactory.newInstance().newSAXParser().getXMLReader();
    reader.setContentHandler(new MyXMLParser());
    reader.parse(new InputSource(inputStream));
  } catch (IOException e) {
    // Deal with IOException
  } catch (SAXException e) {
    // Deal with SAXException
  } catch (ParserConfigurationException e) {
    // Deal with ParserConfigurationException
  }
}

public void startElement(String uri, String localname, String qName,
                         Attributes atts)
{
    if (qName.equals("Error"))
      System.out.println("Error code: " + atts.getValue("code")
        + ", description: " + atts.getValue("description"));
    else if (qName.equals("Records")) {
      currentReqSym = atts.getValue("req_sym");
      currentKey    = atts.getValue("key");
      currentStale  = atts.getValue("stale");
    }
    else if (qName.equals("Field"))
      System.out.println("req_sym: " + currentReqSym
        + ", key: " + currentKey + ", stale: " + currentStale
        + ", id: " + atts.getValue("id")
        + ", name: " + atts.getValue("name")
        + " = " + atts.getValue("value"));
}

private String currentReqSym, currentKey, currentStale;

public void characters(char [] ch, int start, int length)           {}
public void endDocument()                                           {}
public void endElement(String uri, String localname, String qName)  {}
public void endPrefixMapping(String prefix)                         {}
public void ignorableWhitespace(char [] ch, int start, int length)  {}
public void processingInstruction(String target, String data)       {}
public void setDocumentLocator(Locator locator)                     {}
public void skippedEntity(String name)                              {}
public void startDocument()                                         {}
public void startPrefixMapping(String prefix, String uri)           {}

} // class MyXMLParser
```

[10] Applications may also use an XPath and Document object to parse the response, see Appendix B.

Program output

```
req_sym: FDS-USA, key: FDS-USA, stale: , id: 300, name: LAST_1 = 92.29
req_sym: FDS-USA, key: FDS-USA, stale: , id: 302, name: LAST_TIME_1 = 100000
req_sym: FDS-USA, key: FDS-USA, stale: , id: 304, name: LAST_VOL_1 = 21725
req_sym: FDS-USA, key: FDS-USA, stale: , id: 300, name: LAST_1 = 92.3
req_sym: FDS-USA, key: FDS-USA, stale: , id: 302, name: LAST_TIME_1 = 110000
req_sym: FDS-USA, key: FDS-USA, stale: , id: 304, name: LAST_VOL_1 = 42516
req_sym: FDS-USA, key: FDS-USA, stale: , id: 300, name: LAST_1 = 92.81
req_sym: FDS-USA, key: FDS-USA, stale: , id: 302, name: LAST_TIME_1 = 120000
req_sym: FDS-USA, key: FDS-USA, stale: , id: 304, name: LAST_VOL_1 = 23419
```

3 Processing Responses

3.1 Response when Specifying Fields Explicitly

3.1.1 Example XML Response

This request returns the LAST_1, LAST_TIME_1, LAST_VOL_1 fields for the symbols IBM-USA on 09/10/2017 between 10:00 AM and 3:00 PM in one hour intervals:
https://datadirect-beta.factset.com/services/TickHistory?id=IBM-USA&date=20170914&st=100000&et=140000&=&interval=1H&format=xml&fields=last_1,last_time_1,last_vol_1

```
<Response>
   <Request>
      <RequestedSymbol>IBM-USA</RequestedSymbol>
      <RequestedFields>last_1,last_time_1,last_vol_1</RequestedFields>
      <RequestKey>4EDFD73F3E153492</RequestKey>
   </Request>
   <Error code="0" description="" />
   <Records key="IBM-USA">
      <Record>
         <Fields>
            <Field id="300" name="LAST_1" value="185.51" />
            <Field id="302" name="LAST_TIME_1" value="100000" />
            <Field id="304" name="LAST_VOL_1" value="1013024" />
         </Fields>
      </Record>
      <Record>
         <Fields>
            <Field id="300" name="LAST_1" value="185.83" />
            <Field id="302" name="LAST_TIME_1" value="110000" />
            <Field id="304" name="LAST_VOL_1" value="418542" />
         </Fields>
      </Record>
      <Record>
         <Fields>
            <Field id="300" name="LAST_1" value="185.56" />
            <Field id="302" name="LAST_TIME_1" value="120000" />
            <Field id="304" name="LAST_VOL_1" value="419045" />
         </Fields>
      </Record>
      <Record>
         <Fields>
            <Field id="300" name="LAST_1" value="185.72" />
            <Field id="302" name="LAST_TIME_1" value="130000" />
            <Field id="304" name="LAST_VOL_1" value="501591" />
         </Fields>
      </Record>
   </Records>
</Response>
```

3.1.2 CSV Response with Specified Fields

If the response format specifies "csv", a CSV file would be sent.

https://datadirect-beta.factset.com/services/TickHistory?id=IBM-USA&date=20170914&st=100000&et=140000&=&interval=1H&format=csv&fields=last_1,last_time_1,last_vol_1

The URL above produces the following CSV response:

```
LAST_1,LAST_TIME_1,LAST_VOL_1
185.51,100000,1013024
185.83,110000,418542
185.56,120000,419045
185.72,130000,501591
```

3.1.3 XML Response with All Fields

This example contains all the fields for the symbol IBM-USA (since the fields parameter is not present).

https://datadirect-beta.factset.com/services/TickHistory?id=IBM-USA&date=20170914&st=100000&et=100500&=&interval=5M&format=xml


```
<Response> -<Request id="">
            <RequestedSymbol>IBM-USA</RequestedSymbol>

      <RequestedFields>BID_1,BID_VOL_1,BID_EXCH_1,ASK_1,ASK_VOL_1,ASK_EXCH_1,LAST_
1,LAST_DATE_1,LAST_TIME_1,LAST_VOL_1,LAST_EXCH_1,CUM_VOL,VWAP,OPEN_1,HIGH_1,LOW_1,
TRADE_CONDITION,GMT_OFFSET,PRICE_CURRENCY</RequestedFields>
            <RequestKey>541042DE6C0F6E82</RequestKey>
      </Request>
      <Error description="" code="0"/> -<Records stale="" key="IBM-USA"
req_sym="IBM-USA"> -<Record> -<Fields>
                        <Field id="100" value="190.27" name="BID_1"/>
                        <Field id="104" value="1" name="BID_VOL_1"/>
                        <Field id="107" value="11106" name="BID_EXCH_1"/>
                        <Field id="200" value="190.31" name="ASK_1"/>
                        <Field id="204" value="1" name="ASK_VOL_1"/>
                        <Field id="207" value="10050" name="ASK_EXCH_1"/>
                        <Field id="300" value="190.27" name="LAST_1"/>
                        <Field id="301" value="20170909" name="LAST_DATE_1"/>
                        <Field id="302" value="100453990" name="LAST_TIME_1"/>
                        <Field id="304" value="20428" name="LAST_VOL_1"/>
                        <Field id="307" value="11009" name="LAST_EXCH_1"/>
                        <Field id="601" value="355797" name="CUM_VOL"/>
                        <Field id="603" value="190.17" name="VWAP"/>
                        <Field id="710" value="190.02" name="OPEN_1"/>
                        <Field id="720" value="190.3" name="HIGH_1"/>
                        <Field id="723" value="189.98" name="LOW_1"/>
                        <Field id="2709" value="01 2 24" name="TRADE_CONDITION"/>
                        <Field id="2037" value="-240" name="GMT_OFFSET"/>
                        <Field id="2032" value="USD" name="PRICE_CURRENCY"/>
                  </Fields>
            </Record> -<Record> -<Fields>
                        <Field id="100" value="190.28" name="BID_1"/>
                        <Field id="104" value="1" name="BID_VOL_1"/>
                        <Field id="107" value="10050" name="BID_EXCH_1"/>
                        <Field id="200" value="190.35" name="ASK_1"/>
                        <Field id="204" value="1" name="ASK_VOL_1"/>
                        <Field id="207" value="11009" name="ASK_EXCH_1"/>
                        <Field id="300" value="190.28" name="LAST_1"/>
                        <Field id="301" value="20170909" name="LAST_DATE_1"/>
                        <Field id="302" value="100558948" name="LAST_TIME_1"/>
                        <Field id="304" value="2500" name="LAST_VOL_1"/>
                        <Field id="307" value="10050" name="LAST_EXCH_1"/>
                        <Field id="601" value="359361" name="CUM_VOL"/>
                        <Field id="603" value="190.171" name="VWAP"/>
                        <Field id="710" value="190.26" name="OPEN_1"/>
                        <Field id="720" value="190.3" name="HIGH_1"/>
                        <Field id="723" value="190.225" name="LOW_1"/>
                        <Field id="2709" value="01 2 24" name="TRADE_CONDITION"/>
                        <Field id="2037" value="-240" name="GMT_OFFSET"/>
                        <Field id="2032" value="USD" name="PRICE_CURRENCY"/>
```

```
                    </Fields>
                </Record>
            </Records>
</Response>
```

3.1.4 CSV Response with All Fields

This example contains all the fields for the symbol IBM-USA in a CSV response (since the fields variable is not present).

https://datadirect-beta.factset.com/services/TickHistory?id=IBM-USA&date=20170914&st=100000&et=100500&=&interval=5M&format=csv

The URL above produces the following CSV response:

```
BID_1,BID_VOL_1,BID_EXCH_1,ASK_1,ASK_VOL_1,ASK_EXCH_1,LAST_1,LAST_DATE_1,LAST_TIME_1,LAST_VOL_1,LAST_EXCH_1,CUM_VOL,VWAP,OPEN_1,HIGH_1,LOW_1,TRADE_CONDITION,GMT_OFFSET,PRICE_CURRENCY
190.27,1,11106,190.31,1,10050,190.27,20170909,100453990,20428,11009,355797,190.17,190.02,190.3,189.98,01 224,-240,USD
190.28,1,10050,190.35,1,11009,190.28,20170909,100558948,2500,10050,359361,190.171,190.26,190.3,190.225,01 224,-240,USD
```

3.1.5 XML Response Header

Each XML response will have a response header. The header contains the request id, requested symbol, and requested fields. In addition, if a request is invalid an error code along with a description will be sent via the header.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<Response>
   <Request>
      <RequestedSymbol>IBM-USA</RequestedSymbol>
      <RequestedFields />
      <RequestKey>4EDFD79DA863FD66</RequestKey>
   </Request>
   <Error code="0" description="" />
   .
   .
   .
```

3.1.6 XML Response Body

The response body contains the data for all items requested. For each data item, a record will be generated in the output stream. Records contain multiple fields and each field will have a name, id (the field identifier), and value.

```
</Response>
```

```
.
.
.
            </Fields>
            </Record> -<Record> -<Fields>
                        <Field id="100" value="190.28" name="BID_1"/>
                        <Field id="104" value="1" name="BID_VOL_1"/>
                        <Field id="107" value="10050" name="BID_EXCH_1"/>
                        <Field id="200" value="190.35" name="ASK_1"/>
                        <Field id="204" value="1" name="ASK_VOL_1"/>
                        <Field id="207" value="11009" name="ASK_EXCH_1"/>
                        <Field id="300" value="190.28" name="LAST_1"/>
                        <Field id="301" value="20170909" name="LAST_DATE_1"/>
                        <Field id="302" value="100558948" name="LAST_TIME_1"/>
                        <Field id="304" value="2500" name="LAST_VOL_1"/>
                        <Field id="307" value="10050" name="LAST_EXCH_1"/>
                        <Field id="601" value="359361" name="CUM_VOL"/>
                        <Field id="603" value="190.171" name="VWAP"/>
                        <Field id="710" value="190.26" name="OPEN_1"/>
                        <Field id="720" value="190.3" name="HIGH_1"/>
                        <Field id="723" value="190.225" name="LOW_1"/>
                        <Field id="2709" value="01 2 24" name="TRADE_CONDITION"/>
                        <Field id="2037" value="-240" name="GMT_OFFSET"/>
                        <Field id="2032" value="USD" name="PRICE_CURRENCY"/>
                  </Fields>
            </Record>
      </Records>
```

3.2 Errors in the Response

The Exchange Snapshot History Service does not send HTTP errors. Instead, all errors are included in the error element or field values. For XML responses, errors will be contained in the XML header (See section 3.1.5 XML Response Header). CSV responses contain the error in the CSV data (since there is no header).

Error codes are sent in the XML response header or CSV file. The following values are possible:

Error Code	Meaning
0	No Error
401	Access Denied. The user is not authorized to view the data requested. Contact FactSet Consulting Services for assistance.
403	Invalid HTTP URL. Necessary query parameters are missing in the request. The description field will indicate the exact reason.
405	Invalid HTTP method. Either the method is not GET or exceeds the maximum request length (currently set at 5000 bytes). The description field will indicate the exact reason.

3.2.1 Access Denied

If a request asks for data that the user is not entitled to , the following error is generated:

XML Output:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<Response>
    <Request>
        <RequestedSymbol>IBM</RequestedSymbol>
        <RequestedFields></RequestedFields>
        <RequestKey>4EDFE6E3B0FC063B</RequestKey>
    </Request>
    <Error code="401" description="You are not authorized for this data"/>
</Response>
```

CSV Output:

```
Requested Symbols,Requested Fields,Request Key,Error Code,Error Description
IBM,,4EDFE84D0C96E6B7,401,You are not authorized for this data
```

3.2.2 Invalid URL

If there are no symbols in the query string, the following error is generated:

XML Output:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<Response>
    <Request>
        <RequestedSymbol></RequestedSymbol>
        <RequestedFields></RequestedFields>
        <RequestKey>4EDFE2AFF649A595</RequestKey>
    </Request>
    <Error code="403" description="No symbol specified in query string"/>
</Response>
```

CSV Output:

```
Requested Symbols,Requested Fields,Request Key,Error Code,Error Description
,,4EDFD9C74719ACD6,403,No symbol specified in query string
```

3.2.3 Invalid Method

If a request is not a GET or is larger than 5000 bytes, the following error is generated:

XML Output:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<Response>
    <Request>
        <RequestedSymbol></RequestedSymbol>
        <RequestedFields></RequestedFields>
        <RequestKey>4EDFE6E3B0FC063B</RequestKey>
    </Request>
    <Error code="405" description="Only GET method is allowed"/>
</Response>
```

CSV Output:

```
Requested Symbols,Requested Fields,Request Key,Error Code,Error Description
,, 4EDFE6E3B0FC063B,405,Only GET method is allowed
```

3.3 Special Field Values

Special values may be sent to indicate certain types of error conditions. These values are used when attempting to force the response into a matrix for both CSV and XML formats.

Special Field Value	Meaning
#NF#	Not Found. The requested symbol could not be found by the service.
#NA#	Not Applicable. The requested field could not be found for the requested symbol. This usually means that the field is not applicable for this type of record,

3.3.1 Symbol Not Found

This request generates a valid response for the first symbol, but the second symbol is invalid.

https://datadirect-beta.factset.com/services/TickHistory?&date=20170914&st=100000&et=100500&=&format=xml&id=sdfs

XML Output:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<Response>
    <Request>
        <RequestedSymbol>sdfs</RequestedSymbol>
        <RequestedFields></RequestedFields>
        <RequestKey>4EDFE408BCEF56E8</RequestKey>
    </Request>
    <Error code="0" description=""/>
    <Records req_sym="sdfs" key="#NF#" stale="Symbol not found">
        <Record>
            <Fields>
                <Field id="300" name="LAST_1" value="#NF#"/>
                <Field id="710" name="OPEN_1" value="#NF#"/>
                <Field id="720" name="HIGH_1" value="#NF#"/>
                <Field id="723" name="LOW_1" value="#NF#"/>
                <Field id="304" name="LAST_VOL_1" value="#NF#"/>
                <Field id="307" name="LAST_EXCH_1" value="#NF#"/>
                <Field id=”301" name="LAST_DATE_1" value="#NF#"/>
                <Field id="302" name="LAST_TIME_1" value="#NF#"/>
                <Field id="601" name="CUM_VOL" value="#NF#"/>
                <Field id="603" name="VWAP" value="#NF#"/>
            </Fields>
        </Record>
    </Records>
</Response>
```

CSV Output:

```
LAST_1,OPEN_1,HIGH_1,LOW_1,LAST_VOL_1,LAST_EXCH_1,LAST_DATE_1,LAST_TIME
_1,CUM_VOL,VWAP
#NF#,#NF#,#NF#,#NF#,#NF#,#NF#,#NF#,#NF#,#NF#,#NF#
```

4 FactSet Data Link

4.1 Introduction

FactSet Data Link allows individual users to make requests using their terminal permissions. A FactSet Workstation with the FactSet Data Link must be installed for this service to work, and you must be logged into FactSet. A secure token will be generated that needs to be used when requesting data.

Users of the Exchange Snapshot Service will, in most cases, need exchange redistribution agreements in order to share the data internally or externally. The user is also responsible for managing their permissions. With FactSet Data Link, the exchange redistribution agreements are not necessary as you are not allowed to share or distribute market data information outside of the FactSet terminal. In addition, FactSet maintains the individual users' permission maps. The FactSet Data Link response will be in the same format as the Exchange Snapshot Service.

The authentication token is only valid as long as you are logged into FactSet. Once you have logged out, your authentication token will no longer be valid and cannot be used to request data. A new token is not needed on every request, but the token will expire after a certain amount of time, usually 12 hours. Before the token expires, FactSet recommends asking for a new token so there will be no disruption of service when the token expires.

4.2 How FactSet Data Link works

A new service is setup when you subscribe to the FactSet Data Link product. This new service will have a new FactSet OnDemand username, which will be different from your regular FactSet username. Each individual user needs to be permissioned by FactSet in order to use the FactSet Data Link. Before you make a request, the client application must use FactSet Workstation to generate an authentication token. This token is then included in the web request and allows FactSet to permission each request based on your individual permissions.

4.3 Differences between FactSet Data Link and the Enterprise Service

The responses between FactSet Data Link and the Enterprise Snapshot are exactly the same; however, FactSet Data Link requests require some additional steps to be taken.

Before making a FactSet Data Link request, a COM call will need to be made to generate an authentication token. The FDSW COM object will be used to call the RunApplication function. Its parameters are "username" and "env" (environment). The username is the username of the OnDemand service, not your FactSet username, and will be provided when the FactSet Data Link service is setup.

> Here are example parameters that will be passed into the RunApplication call:
>
> USERNAME=CompanyX_123456_Services
>
> Note: USERNAME will be the new OnDemand username that provides data to FactSet Data Link, not your current FactSet username.

Calling this function will start FactSet and prompt a login if not already present. Once logged in, the authentication token will be negotiated with a back-end service and a string containing the 16 printable character auth token will be returned. If there is an error, a string will be returned explaining the problem (see section 4.4 RunApplication (COM Function) Return Values). The auth token should then be saved and added to all subsequent requests in the HTTP header X-DataDirect-Auth-Token. This token will be valid for up to 12 hours before expiring. Also, if the terminal is logged out, the token will be invalidated. Since negotiating a new auth token is a relatively expensive operation, it is recommended that the COM function only be called when first logging in and if 12 hours has elapsed. While it would work to negotiate a new auth token for every request, it would unnecessarily delay responses from the service.

Section 4.6 Example Code for HTTP GET Request for FactSet Data Link contains example code for an HTTP GET request through FactSet Data Link. The code that has been added from the Enterprise Snapshot Service example is in bold (except the section that makes the COM call and gets an authentication token). Besides adding the extra HTTP header, two more "using" lines have

been added. The new section that gets the auth token should be split out so it doesn't get called for every request, but this works as a simple example.

4.4 RunApplication (COM Function) Return Values

This table lists the possible string return values from the COM function RunApplication:

String Returned from RunApplication	Meaning
TOKEN={string consisting of 16 printable characters}	Success. Parse out the 16 character authentication token for later use.
ERROR_UNAUTHORIZED	Error. This user does not have permission to use the TickHistory service.
ERROR_MISSING_PARAMETER	Error. A required parameter to RunApplication is missing, such as "USERNAME".
ERROR_GENERATING_AUTH_TOKEN	Error. An internal error occurred while generating the authentication token. Please try the COM call again.

4.5 Moving to Production from Beta

One change to the below sample code will need to be made for the move to production.

1. The URL should be changed from "https://**datadirect-beta**.factset.com/services/TickHistory" to "https://**datadirect**.factset.com/services/TickHistory" to hit FactSet's production FactSet OnDemand servers.

4.6 Example Code for HTTP GET Request for FactSet Data Link

```
using System.IO;
using System.IO.Compression;
using System.Net;
using System.Xml;
using System.Reflection;    // This is needed for BindingFlags
using System;               // This is needed for Type and Activator

class MyXMLParser
{

[System.STAThread]
static void Main(string [] args)
{
  // Make a COM call to RunApplication to get an MDL auth token
  // This call is expensive and should only be run appox every 12 hours
  Type fds_obj_type = Type.GetTypeFromProgID("FactSet.FactSet_API");
  object fds_obj = Activator.CreateInstance(fds_obj_type);
  string[] runapp_params = new string[] { "FDSA DataDirect Token",
                "username=USERNAME_123456_SERVICES"};
  string runapp_rtn = (string)fds_obj_type.InvokeMember(
                      "RunApplication", BindingFlags.InvokeMethod,
                      null, fds_obj, runapp_params);
  string auth_token = "";
  if (runapp_rtn.StartsWith("TOKEN="))
    auth_token = runapp_rtn.Substring(6);  // Parse out the auth token
  else
  {
    System.Console.WriteLine("Auth Token Gen Error: " + runapp_rtn);
    return;
  }
  .
  .
  .
  // Make a normal snapshot request, but append the returned
  // authentication token in the header X-DataDirect-Auth-Token
  HttpWebRequest req = (HttpWebRequest)WebRequest.Create(
      "https://datadirect-beta.factset.com/services/TickHistory?" +
      "id=FDS-USA&format=xml" +
      "&date=20170914&fields=LAST_1,LAST_TIME_1,LAST_VOL_1" +
      "&interval=1H&st=100000&et=130000");
  req.KeepAlive = false;
  req.Headers.Add("Accept-Encoding", "deflate, gzip");
  req.Headers.Add("Authorization", "Basic AaBbCcDdEeFfGgHhIi1234==");
  req.Headers.Add("X-DataDirect-Auth-Token", auth_token);

  HttpWebResponse rsp = (HttpWebResponse)req.GetResponse();
  Stream stream = rsp.GetResponseStream();

  if (rsp.ContentEncoding.Equals("deflate"))
    stream = new DeflateStream(stream, CompressionMode.Decompress);
  else if (rsp.ContentEncoding.Equals("gzip"))
    stream = new GZipStream(stream, CompressionMode.Decompress);
  .
  .
```

5 Appendix

5.1 Appendix A: Using C#'s XMLDocument

This example code illustrates parsing an XML Response using C#'s XMLDocument. It loads the XML into an XMLDocument instance and then uses XPath to select nodes of the XMLDocument and print out the relevant data.

```
  .
  .
  // Get XML Response using GET (see 2.4.1)
  .
  .
  XmlTextReader reader = new XmlTextReader(stream);
  reader.WhitespaceHandling = WhitespaceHandling.None;

  XmlDocument doc = new XmlDocument();
  doc.Load(reader);

  foreach (XmlNode node in doc.SelectNodes("/Response/Error"))
  {
    System.Console.WriteLine(node.Name);
    printAttributes(node);
    System.Console.WriteLine();
  }

  foreach (XmlNode node in
           doc.SelectNodes("/Response/Records/Record"))
  {
    System.Console.WriteLine(node.Name);

    foreach (XmlNode filedNode in node.SelectNodes("Field"))
      printAttributes(fieldNode);

    System.Console.WriteLine();
  }
}

static void printAttributes(XmlNode node)
{
  foreach (XmlAttribute att in node.Attributes)
    System.Console.Write(att.Name + ": " + att.Value + ", ");
  System.Console.WriteLine();
}

} // class MyXMLParser
```

5.2 Appendix B: Using Java's Document Class

This example code illustrates parsing an XML Response using Java's Document Class. It loads the XML into a Document instance and then uses XPath to select nodes of the Document and print out the relevant data[11].

```
import java.io.*;
import java.net.URL;
import javax.net.ssl.HttpsURLConnection;
import javax.xml.parsers.*;
import javax.xml.xpath.*;
import org.w3c.dom.*;
import org.xml.sax.*;

public class MyDOMXpathParser {

public static void main(String[] args)
{
  try {
    // Get XML Response using GET (see 2.4.2)

    // Parse the XML into a DOM Document
    DocumentBuilderFactory docFactory =
      DocumentBuilderFactory.newInstance();
    docFactory.setNamespaceAware( true );
    DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
    Document doc = docBuilder.parse(inputStream);

    // Set up an XPath for easy access of XML structure
    XPath xp = XPathFactory.newInstance().newXPath();

    // Go through the XML Response and print out the relevant data
    NodeList nodes = (NodeList) xp.evaluate("/Response/Error",
        doc.getDocumentElement(), XPathConstants.NODESET);
    for ( int i = 0; i < nodes.getLength(); ++i)
    {
      Node errorNode = nodes.item(i);
      System.out.println(errorNode.getNodeName());
      printAttributes(errorNode);
      System.out.println();
    }

    nodes = (NodeList) xp.evaluate("/Response/Records/Record",
        doc.getDocumentElement(), XPathConstants.NODESET);
    for ( int i = 0; i < nodes.getLength(); ++i)
    {
      Node recordNode = nodes.item(i);
      System.out.println(recordNode.getNodeName());

      NodeList fieldNodes = (NodeList) xp.evaluate("Field",
          recordNode, XPathConstants.NODESET);
      for ( int k = 0; k < fieldNodes.getLength(); ++k)
      {
        Node fieldNode = fieldNodes.item(k);
        printAttributes(fieldNode);
      }
```

[11] This method of parsing does not use event-driven callbacks, and thus the structure is modified accordingly.

```
      System.out.println();
    }
  } catch (IOException e) {
    // Deal with IOException
  } catch (XPathExpressionException e) {
    // Deal with XPathExpressionException
  } catch (ParserConfigurationException e) {
    // Deal with ParserConfigurationException
  } catch (SAXException e) {
    // Deal with SAXException
  }
}

// Helper function that prints out all attributes of a node
private static void printAttributes(Node node)
{
  NamedNodeMap atts = node.getAttributes();
  for ( int i = 0; i < atts.getLength(); ++i)
  {
    Node att = atts.item(i);
    System.out.print(att.getNodeName() + ": " + att.getNodeValue()
        + ", ");
  }
  System.out.println();
}

} // class MyXMLParser
```