```
//**
//** This Example shows how to Retrieve reference data/Bulk reference data using BLP
//     Server ** API

//*/

using Event = Bloomberglp.Blpapi.Event;
using Element = Bloomberglp.Blpapi.Element;
using InvalidRequestException = Bloomberglp.Blpapi.InvalidRequestException;
using Message = Bloomberglp.Blpapi.Message;
using Name = Bloomberglp.Blpapi.Name;
using Request = Bloomberglp.Blpapi.Request;
using Service = Bloomberglp.Blpapi.Service;
using Session = Bloomberglp.Blpapi.Session;
using Datatype = Bloomberglp.Blpapi.Schema.Datatype;

using ArrayList = System.Collections.ArrayList;

using System.Text.RegularExpressions;
using System;
using System.IO;
using System.Collections;
using System.Collections.Generic;

namespace Bloomberglp.Blpapi.Examples
{
    public class BulkRefDataExample
    {
        private static readonly Name SECURITY_DATA = new Name("securityData");
        private static readonly Name SECURITY = new Name("security");
        private static readonly Name FIELD_DATA = new Name("fieldData");
        private static readonly Name RESPONSE_ERROR = new Name("responseError");
        private static readonly Name SECURITY_ERROR = new Name("securityError");
        private static readonly Name FIELD_EXCEPTIONS = new Name("fieldExceptions");
        private static readonly Name FIELD_ID = new Name("fieldId");
        private static readonly Name ERROR_INFO = new Name("errorInfo");
        private static readonly Name CATEGORY = new Name("category");
        private static readonly Name MESSAGE = new Name("message");

        private string d_host;
        private int d_port;
        private ArrayList d_securities;
        private ArrayList d_fields;

        private static string xRootDir = "";
        private static StreamReader f_r;// = new
StreamReader(@"T:\SmallUpdate\xListDVD.txt");
        private static StreamWriter f_w;// = new
StreamWriter(@"T:\SmallUpdate\xDVD_HIST_ALL.txt");

        //private static StreamWriter f_w = new
StreamWriter(@"T:\SmallUpdate\xStaticData.txt");

        private static string xDeclaredDate = "";
        private static string xExDate = "";
        private static string xRecordDate = "";
        private static string xPayableDate = "";
        private static double xDividendAmount = 0;
        private static string xDividendFrequency = "";
        private static string xDividendType = "";

        private static string xYearPeriod = "";
        private static string xAnnouncementDate = "";
        private static string xAnnouncementTime = "";
        private static double xEarningsEPS = 0;
        private static double xComparableEPS = 0;
        private static double xEstimateEPS = 0;

        private static string[] xTempArray2;
        private static string xTempString = "";
        private static int xCountSent = 0;
        private static int xCountProcessed = 0;
        private static string xCUSIP = "";

        private static int x10Pct = 0;
        private static int x20Pct = 0;
        private static int x30Pct = 0;
        private static int x40Pct = 0;
        private static int x50Pct = 0;
        private static int x60Pct = 0;
        private static int x70Pct = 0;
        private static int x80Pct = 0;
        private static int x90Pct = 0;

        private static Dictionary<string, int> xDictID = new Dictionary<string, int>();

        private static System.DateTime xTempDate = System.DateTime.Now;

        public static void Main(string[] args)
        {
            System.Console.WriteLine("Reference Data/Bulk Reference Data Example");

            //xRootDir = @"T:\SmallUpdate\";
            xRootDir = AppDomain.CurrentDomain.BaseDirectory; //this constain "\"!!
            //xRootDir = @"C:\GU\EarningsEstimatesSurpises\";

            f_r = new StreamReader(xRootDir + "xListOfStocks.txt");
            f_w = new StreamWriter(xRootDir + "xEarnings.txt");


f_w.WriteLine("ticker,announcement_period,announcement_date,announcement_time,earnings_ep
s,comparable_eps,estimate_eps");

            BulkRefDataExample example = new BulkRefDataExample();
            example.run(args);

            System.Console.WriteLine("Completed.");
            //System.Console.WriteLine("Press ENTER to quit");
            //System.Console.Read();
        }

        /// <summary>
        /// Constructor
        /// </summary>
        public BulkRefDataExample()
        {
```

```
        d_host = "localhost";
        d_port = 8194;
        d_securities = new ArrayList();
        d_fields = new ArrayList();
    }

    /// <summary>
    /// Read command line arguments,
    /// Establish a Session
    /// Identify and Open refdata Service
    /// Send ReferenceDataRequest to the Service
    /// Event Loop and Response Processing
    /// </summary>
    /// <param name="args"></param>
    private void run(string[] args)
    {
        if (!parseCommandLine(args)) return;

        SessionOptions sessionOptions = new SessionOptions();
        sessionOptions.ServerHost = d_host;
        sessionOptions.ServerPort = d_port;

        System.Console.WriteLine("Connecting to " + d_host + ":" + d_port);
        Session session = new Session(sessionOptions);
        bool sessionStarted = session.Start();
        if (!sessionStarted)
        {
            System.Console.Error.WriteLine("Failed to start session.");
            return;
        }
        if (!session.OpenService("//blp/refdata"))
        {
            System.Console.Error.WriteLine("Failed to open //blp/refdata");
            return;
        }

        //making requests...

        // handle default arguments
        if (d_securities.Count == 0)
        {
            //d_securities.Add("WOR Equity");
            //d_securities.Add("CAC Index");
            //d_securities.Add("CTCM Equity");
            //d_securities.Add("DELL Equity");
            //d_securities.Add(@"/cusip/12642X10");
            //d_securities.Add(@"/cusip/24702R10");
            while ((xTempString = f_r.ReadLine()) != null)
            {
                //d_securities.Add(xTempString + " Equity");
                //xCountSent = xCountSent + 1;
                xTempArray2 = Regex.Split(xTempString, ",");
                if (xTempArray2.Length == 2)
                {
                    xCUSIP = xTempArray2[1];
                    //d_securities.Add(@"/cusip/" + xCUSIP);    //using CUSIP
                    d_securities.Add(xCUSIP + " Equity");
                    xCountSent = xCountSent + 1;
                    }
                }
            }

            if (d_fields.Count == 0)
            {
                //d_fields.Add("INDX_MWEIGHT");
                //d_fields.Add("DVD_HIST_ALL");
                //d_fields.Add("ID_CUSIP_8_CHR");
                d_fields.Add("EARN_ANN_DT_TIME_HIST_WITH_EPS");
                //d_fields.Add("TICKER");
                //d_fields.Add("NAME");
                //d_fields.Add("GICS_SECTOR_NAME");
                //d_fields.Add("SECURITY_TYP2");
                //d_fields.Add("DVD_CRNCY");

            }
            //

            try
            {
                sendRefDataRequest(session);
                System.Console.WriteLine("Total of " + xCountSent + " requests are
made...");
            }
            catch (InvalidRequestException e)
            {
                System.Console.WriteLine(e.ToString());
            }

            // wait for events from session.
            eventLoop(session);

            session.Stop();
        }

        /// <summary>
        /// Polls for an event or a message in an event loop
        /// & Processes the event generated
        /// </summary>
        /// <param name="session"></param>
        private void eventLoop(Session session)
        {
            bool done = false;
            while (!done)
            {
                Event eventObj = session.NextEvent();
                if (eventObj.Type == Event.EventType.PARTIAL_RESPONSE)
                {
                    //System.Console.WriteLine("Processing Partial Response");
                    processResponseEvent(eventObj);
                }
                else if (eventObj.Type == Event.EventType.RESPONSE)
                {
                    //System.Console.WriteLine("Processing Response");
                    processResponseEvent(eventObj);
                    done = true;
                }
            else
            {
                foreach (Message msg in eventObj)
                {
                    //System.Console.WriteLine(msg.AsElement);
                    if (eventObj.Type == Event.EventType.SESSION_STATUS)
                    {
                        if (msg.MessageType.Equals("SessionTerminated"))
                        {
                            done = true;
                        }
                    }
                }
            }
        }
    }

    /// <summary>
    /// Function to handle response event
    /// </summary>
    /// <param name="eventObj"></param>
    private void processResponseEvent(Event eventObj)
    {
        foreach (Message msg in eventObj)
        {
            if (msg.HasElement(RESPONSE_ERROR))
            {
                printErrorInfo("REQUEST FAILED: ", msg.GetElement(RESPONSE_ERROR));
                continue;
            }

            Element securities = msg.GetElement(SECURITY_DATA);
            int numSecurities = securities.NumValues;
            //System.Console.WriteLine("\nProcessing " + numSecurities
            //                            + " securities:");
            for (int secCnt = 0; secCnt < numSecurities; ++secCnt)
            {
                Element security = securities.GetValueAsElement(secCnt);
                string ticker = security.GetElementAsString(SECURITY);
                //string xCUSIP = ""; //in global now!
                string[] xTempArray = Regex.Split(ticker, @" ");
                ticker = xTempArray[0];

                //if (xTempArray.Length == 3)
                //{
                //    xCUSIP = xTempArray[2];
                //}

                //System.Console.WriteLine("\nTicker: " + ticker);
                if (security.HasElement("securityError"))
                {
                    printErrorInfo("\tSECURITY FAILED: ",
                        security.GetElement(SECURITY_ERROR));
                    continue;
                }

                Element fields = security.GetElement(FIELD_DATA);
                if (fields.NumElements > 0)
```

```
                    {
                        //System.Console.WriteLine("FIELD\t\tVALUE");
                        //System.Console.WriteLine("-----\t\t-----");
                        int numElements = fields.NumElements;
                        for (int eleCtr = 0; eleCtr < numElements; ++eleCtr)
                        {
                            Element field = fields.GetElement(eleCtr);
                            // Checking if the field is Bulk field
                            if (field.Datatype == Datatype.SEQUENCE)
                            {
                                //processBulkField(field);

                                //processing BULK data BDS() below!!!!
                                //System.Console.WriteLine("\n" + field.Name);
                                // Get the total number of Bulk data points
                                int numofBulkValues = field.NumValues;
                                for (int bvCtr = 0; bvCtr < numofBulkValues; bvCtr++)
                                {
                                    Element bulkElement = field.GetValueAsElement(bvCtr);
                                    // Get the number of sub fields for each bulk data
element
                                    int numofBulkElements = bulkElement.NumElements;
                                    xDeclaredDate = "";
                                    xExDate = "";
                                    xRecordDate = "";
                                    xPayableDate = "";
                                    xDividendAmount = 0;
                                    xDividendFrequency = "";
                                    xDividendType = "";

                                    // Read each field in Bulk data
                                    for (int beCtr = 0; beCtr < numofBulkElements;
beCtr++)
                                    {
                                        Element elem = bulkElement.GetElement(beCtr);
                                        //System.Console.WriteLine("\t\t" + elem.Name + "
= "
                                        //              + elem.GetValueAsString());

                                        xTempString =
elem.Name.ToString().Trim().ToUpper();
                                        switch (xTempString)
                                        {
                                            case "YEAR/PERIOD":
                                                try
                                                {
                                                    xYearPeriod =
elem.GetValueAsString();
                                                }
                                                catch
                                                {
                                                    //do nothing
                                                }
                                                break;
                                            case "ANNOUNCEMENT DATE":
                                                try
                                                {
                                                    xAnnouncementDate = 
elem.GetValueAsDate().ToSystemDateTime().ToShortDateString();
                                                }
                                                catch
                                                {
                                                    //do nothing
                                                }
                                                break;
                                            case "ANNOUNCEMENT TIME":
                                                try
                                                {
                                                    xAnnouncementTime = 
elem.GetValueAsString();
                                                }
                                                catch
                                                {
                                                    //do nothing
                                                }
                                                break;
                                            case "EARNINGS EPS":
                                                try
                                                {
                                                    xEarningsEPS = 
elem.GetValueAsFloat64(); ;
                                                }
                                                catch
                                                {
                                                    //do nothing
                                                }
                                                break;
                                            case "COMPARABLE EPS":
                                                try
                                                {
                                                    xComparableEPS = 
elem.GetValueAsFloat64(); ;
                                                }
                                                catch
                                                {
                                                    //do nothing
                                                }
                                                break;
                                            case "ESTIMATE EPS":
                                                try
                                                {
                                                    xEstimateEPS = 
elem.GetValueAsFloat64(); ;
                                                }
                                                catch
                                                {
                                                    //do nothing
                                                }
                                                break;
                                            //dividends....
                                            case "DECLARED DATE":
                                                try
                                                {
                                                    xDeclaredDate = 
elem.GetValueAsDate().ToSystemDateTime().ToShortDateString();
```

```
                                                }
                                                catch
                                                {
                                                    //do nothing
                                                }
                                                break;

                                            case "EX-DATE":
                                                try
                                                {
                                                    xExDate =
elem.GetValueAsDate().ToSystemDateTime().ToShortDateString();
                                                }
                                                catch
                                                {
                                                    //do nothing
                                                }
                                                break;
                                            case "RECORD DATE":
                                                try
                                                {
                                                    xRecordDate =
elem.GetValueAsDate().ToSystemDateTime().ToShortDateString();
                                                }
                                                catch
                                                {
                                                    //do nothing
                                                }
                                                break;
                                            case "PAYABLE DATE":
                                                try
                                                {
                                                    xPayableDate =
elem.GetValueAsDate().ToSystemDateTime().ToShortDateString();
                                                }
                                                catch
                                                {
                                                    //do nothing
                                                }
                                                break;
                                            case "DIVIDEND AMOUNT":
                                                try
                                                {
                                                    xDividendAmount =
elem.GetValueAsFloat64(); ;
                                                }
                                                catch
                                                {
                                                    //do nothing
                                                }
                                                break;
                                            case "DIVIDEND FREQUENCY":
                                                try
                                                {
                                                    xDividendFrequency =
elem.GetValueAsString();
                                                }
                                                catch
```

```
                                                            {
                                                                //do nothing
                                                            }
                                                            break;
                                                        case "DIVIDEND TYPE":
                                                            try
                                                            {
                                                                xDividendType = 
elem.GetValueAsString();
                                                            }
                                                            catch
                                                            {
                                                                //do nothing
                                                            }
                                                            break;

                                                        //default:
                                                        //    return false;
                                                    }
                                                    //

                                                }
                                                //System.Console.WriteLine("one line is 
processed....");
                                                string xEarningsEPSString = "";
                                                string xComparableEPSString = "";
                                                string xEstimateEPSString = "";

                                                if (xEarningsEPS != 0.0 && Math.Abs(xEarningsEPS * 
1.0E+14) < 1000.0)
                                                {
                                                    xEarningsEPSString = "";
                                                }
                                                else
                                                {
                                                    xEarningsEPSString = xEarningsEPS.ToString();
                                                }
                                                    //

                                                if (xComparableEPS != 0.0 && Math.Abs(xComparableEPS 
* 1.0E+14) < 1000.0)
                                                {
                                                    xComparableEPSString = "";
                                                }
                                                else
                                                {
                                                    xComparableEPSString = xComparableEPS.ToString();
                                                }
                                                    //
                                                if (xEstimateEPS != 0.0 && Math.Abs(xEstimateEPS * 
1.0E+14) < 1000.0)
                                                {
                                                    xEstimateEPSString = "";
                                                }
                                                else
                                                {
                                                    xEstimateEPSString = xEstimateEPS.ToString();
                                                }
```

```
                                    //

                                //xTempString = ticker.Replace(@"/cusip/", "") + ","
+ xYearPeriod + "," + xAnnouncementDate + "," + xAnnouncementTime + ","
                                //       + xEarningsEPS + "," + xComparableEPS + ","
+ xEstimateEPS;

                                xTempString = ticker.Replace(@"/cusip/", "") + "," +
xYearPeriod + "," + xAnnouncementDate + "," + xAnnouncementTime + ","
                                       + xEarningsEPSString + "," +
xComparableEPSString + "," + xEstimateEPSString;


                                //xTempString = xCUSIP + "," + xDeclaredDate + "," +
xExDate + "," + xRecordDate + ","
                                //    + xPayableDate + "," + xDividendAmount + "," +
xDividendFrequency + "," + xDividendType;
                                f_w.WriteLine(xTempString);
                                f_w.Flush();
                            }
                        }
                        else
                        {
                            //processRefField(field);
                            //processing static data:BDP()....

                            //System.Console.WriteLine(field.Name + "\t\t"
                            //    + field.GetValueAsString());

                        }
                        //
                        //System.Console.WriteLine("one secirity is processed....");
                        try
                        {
                            xDictID.Add(xCUSIP, xCountProcessed);
                            xCountProcessed = xCountProcessed + 1;
                            //System.Console.WriteLine(xCountProcessed + " secirity
is processed....");
                            if (Convert.ToDouble(xCountProcessed) /
Convert.ToDouble(xCountSent) > 0.9 && x90Pct == 0)
                            {
                                Console.WriteLine("90% completed");
                                x90Pct = 1;
                            }
                            else if (Convert.ToDouble(xCountProcessed) /
Convert.ToDouble(xCountSent) > 0.8 && x80Pct == 0)
                            {
                                Console.WriteLine("80% completed");
                                x80Pct = 1;
                            }
                            else if (Convert.ToDouble(xCountProcessed) /
Convert.ToDouble(xCountSent) > 0.7 && x70Pct == 0)
                            {
                                Console.WriteLine("70% completed");
                                x70Pct = 1;
                            }
                            else if (Convert.ToDouble(xCountProcessed) /
Convert.ToDouble(xCountSent) > 0.6 && x60Pct == 0)
```

```
                                {
                                    Console.WriteLine("60% completed");
                                    x60Pct = 1;
                                }
                                else if (Convert.ToDouble(xCountProcessed) /
Convert.ToDouble(xCountSent) > 0.5 && x50Pct == 0)
                                {
                                    Console.WriteLine("50% completed");
                                    x50Pct = 1;
                                }
                                else if (Convert.ToDouble(xCountProcessed) /
Convert.ToDouble(xCountSent) > 0.4 && x40Pct == 0)
                                {
                                    Console.WriteLine("40% completed");
                                    x40Pct = 1;
                                }
                                else if (Convert.ToDouble(xCountProcessed) /
Convert.ToDouble(xCountSent) > 0.3 && x30Pct == 0)
                                {
                                    Console.WriteLine("30% completed");
                                    x30Pct = 1;
                                }
                                else if (Convert.ToDouble(xCountProcessed) /
Convert.ToDouble(xCountSent) > 0.2 && x20Pct == 0)
                                {
                                    Console.WriteLine("20% completed");
                                    x20Pct = 1;
                                }
                                else if (Convert.ToDouble(xCountProcessed) /
Convert.ToDouble(xCountSent) > 0.1 && x10Pct == 0)
                                {
                                    Console.WriteLine("10% completed");
                                    x10Pct = 1;
                                }
                            }
                            catch
                            {
                                //could be partial msg...so one same cusip could come
with several msgs!!!
                            }
                        }
                    }

                    //System.Console.WriteLine("");
                    Element fieldExceptions = security.GetElement(FIELD_EXCEPTIONS);
                    if (fieldExceptions.NumValues > 0)
                    {
                        System.Console.WriteLine("FIELD\t\tEXCEPTION");
                        System.Console.WriteLine("-----\t\t---------");
                        for (int k = 0; k < fieldExceptions.NumValues; ++k)
                        {
                            Element fieldException =
                                fieldExceptions.GetValueAsElement(k);
                            printErrorInfo(fieldException.GetElementAsString(FIELD_ID)
                                + "\t\t", fieldException.GetElement(ERROR_INFO));
                        }
                    }
                }
```

```
    }
}

/// <summary>
/// Read the reference bulk field contents
/// </summary>
/// <param name="refBulkField"></param>
private void processBulkField(Element refBulkField)
{
    System.Console.WriteLine("\n" + refBulkField.Name);
    // Get the total number of Bulk data points
    int numofBulkValues = refBulkField.NumValues;
    for (int bvCtr = 0; bvCtr < numofBulkValues; bvCtr++)
    {
        Element bulkElement = refBulkField.GetValueAsElement(bvCtr);
        // Get the number of sub fields for each bulk data element
        int numofBulkElements = bulkElement.NumElements;
        // Read each field in Bulk data
        for (int beCtr = 0; beCtr < numofBulkElements; beCtr++)
        {
            Element elem = bulkElement.GetElement(beCtr);
            System.Console.WriteLine("\t\t" + elem.Name + " = "
                                    + elem.GetValueAsString());
        }
        System.Console.WriteLine("one line is processed....");
    }
    System.Console.WriteLine("one secirity is processed....");
}

/// <summary>
/// Read the reference field contents
/// </summary>
/// <param name="reffield"></param>
private void processRefField(Element reffield)
{
    System.Console.WriteLine(reffield.Name + "\t\t"
                            + reffield.GetValueAsString());

}

/// <summary>
/// Function to create and send ReferenceDataRequest
/// </summary>
/// <param name="session"></param>
private void sendRefDataRequest(Session session)
{
    Service refDataService = session.GetService("//blp/refdata");
    Request request = refDataService.CreateRequest("ReferenceDataRequest");

    // Add securities to request
    Element securities = request.GetElement("securities");

    for (int i = 0; i < d_securities.Count; ++i)
    {
        securities.AppendValue((string)d_securities[i]);
    }

    // Add fields to request
```

```
        Element fields = request.GetElement("fields");
        for (int i = 0; i < d_fields.Count; ++i)
        {
            fields.AppendValue((string)d_fields[i]);
        }

        //System.Console.WriteLine("Sending Request: " + request);
        session.SendRequest(request, null);
    }

    /// <summary>
    /// Parses the command line arguments
    /// </summary>
    /// <param name="args"></param>
    /// <returns></returns>
    private bool parseCommandLine(string[] args)
    {
        for (int i = 0; i < args.Length; ++i)
        {
            if (string.Compare(args[i], "-s", true) == 0)
            {
                d_securities.Add(args[i + 1]);
            }
            else if (string.Compare(args[i], "-f", true) == 0)
            {
                d_fields.Add(args[i + 1]);
            }
            else if (string.Compare(args[i], "-ip", true) == 0)
            {
                d_host = args[i + 1];
            }
            else if (string.Compare(args[i], "-p", true) == 0)
            {
                d_port = int.Parse(args[i + 1]);
            }
            else if (string.Compare(args[i], "-h", true) == 0)
            {
                printUsage();
                return false;
            }
        }

        ////// handle default arguments
        ////if (d_securities.Count == 0)
        ////{
        ////    //d_securities.Add("CAC Index");
        ////    //d_securities.Add("CTCM Equity");
        ////    //d_securities.Add("DELL Equity");
        ////    d_securities.Add(@"/cusip/12642X10");
        ////    d_securities.Add(@"/cusip/24702R10");



        ////}

        ////if (d_fields.Count == 0)
        ////{
        ////    //d_fields.Add("INDX_MWEIGHT");
```

```
            ////    d_fields.Add("DVD_HIST_ALL");
            ////    //d_fields.Add("ID_CUSIP_8_CHR");

            ////    //d_fields.Add("TICKER");
            ////    //d_fields.Add("NAME");
            ////    //d_fields.Add("GICS_SECTOR_NAME");
            ////    //d_fields.Add("SECURITY_TYP2");
            ////    //d_fields.Add("DVD_CRNCY");

            ////}

            return true;
        }

        /// <summary>
        /// Prints error information
        /// </summary>
        /// <param name="leadingStr"></param>
        /// <param name="errorInfo"></param>
        private void printErrorInfo(string leadingStr, Element errorInfo)
        {
            System.Console.WriteLine(leadingStr + errorInfo.GetElementAsString(CATEGORY)
+
                " (" + errorInfo.GetElementAsString(MESSAGE) + ")");
        }

        /// <summary>
        /// Print usage of the Program
        /// </summary>
        private void printUsage()
        {
            System.Console.WriteLine("Usage:");
            System.Console.WriteLine("    Retrieve reference data/Bulk reference"
                              + " data using Server API");
            System.Console.WriteLine("      [-s         <security   = CAC Index>");
            System.Console.WriteLine("      [-f         <field      = INDX_MWEIGHT>");
            System.Console.WriteLine("      [-ip        <ipAddress  = localhost>");
            System.Console.WriteLine("      [-p         <tcpPort    = 8194>");
        }
    }
}
```