NASDAQ Data-On-Demand Developmental Tips

CONTENTS

I. MASS DOWNLOAD FUNCTIONALITY

WHAT IS THE MASS DOWNLOAD FUNCTIONALITY?

Mass Download allows firms to run bulk historical stock tick data queries for integration into application or databases.

How it works:

- Simply submit an online request through your Data-On-Demand Premier subscription. (Note: You can create Mass Download requests using an API as well).
- NASDAQ OMX systems will run and create the data set you requested. Note: While this is idea for mass bulk downloads, this service can also be used for smaller requests.
- You will receive a link to the dataset which allows for easy manual or webservice download. In addition, if your firm has a cloud in place, NASDAQ OMX can automatically send the data directly to that location.
- As an added feature, firms can schedule nightly recurring reports that will automatically download for viewing as soon as data is available.

HOW CAN I SUBSCRIBE TO THE MASS DOWNLOAD FUNCTIONALITY?

To access the Mass Download functionality, you must be a Premier Subscriber. The Mass Download functionality is available to Premier Subscribers only at no additional cost.

For more information on the Mass Download functionality and Premier solution, please email mailto:datasales@nasdaqomx.comor call 301-978-5307 Option #1.

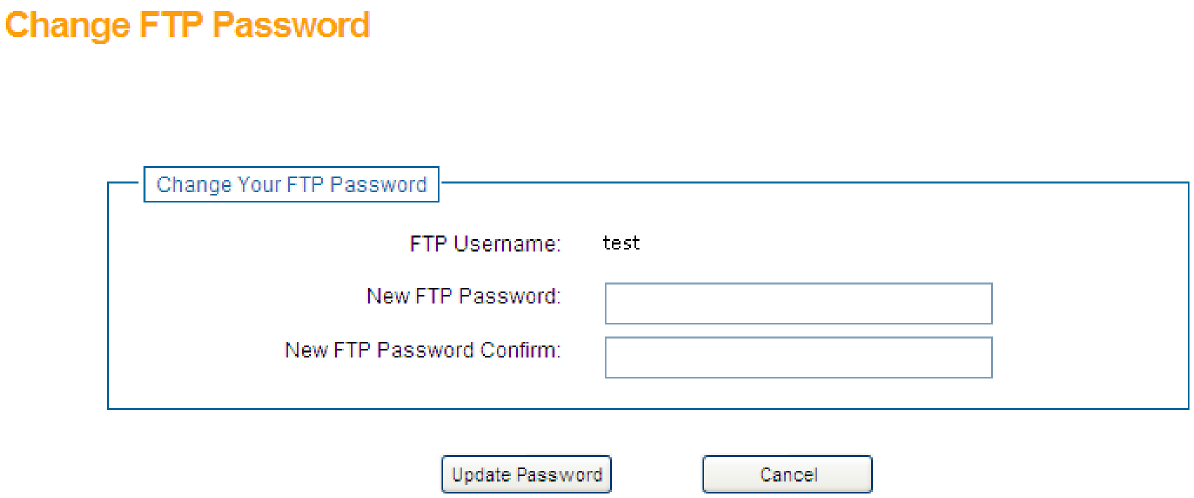
HOW DO I MAKE REQUESTS?

You can make requests using the following methods:

- Website - Simply log in at nasdaqdod.com and go to the Mass Download page. Select the operation you are interested in and choose the parameters.
- Web Services - Customers wanting to access Mass Download through a web service API, please visit www.nasdaqdod.com/NASDAQMassDownload.asmx for more information and detailed documentation.

ONCE I SUBSCRIBE TO PREMIER MEMBERSHIP, HOW DO I ACCESS MASS DOWNLOAD IF I DO NOT HAVE ACCESS TO AN API?

1. Your FTP Username will automatically be created for you. Once you log in to NASDAQ Data-On-Demand with your credentials (Please note: The Website Username/Passwordwill not be the same as your FTP Username/Password), go to My Account, select Updated My FTP Account. Note your FTP username below:



Change FTP Password

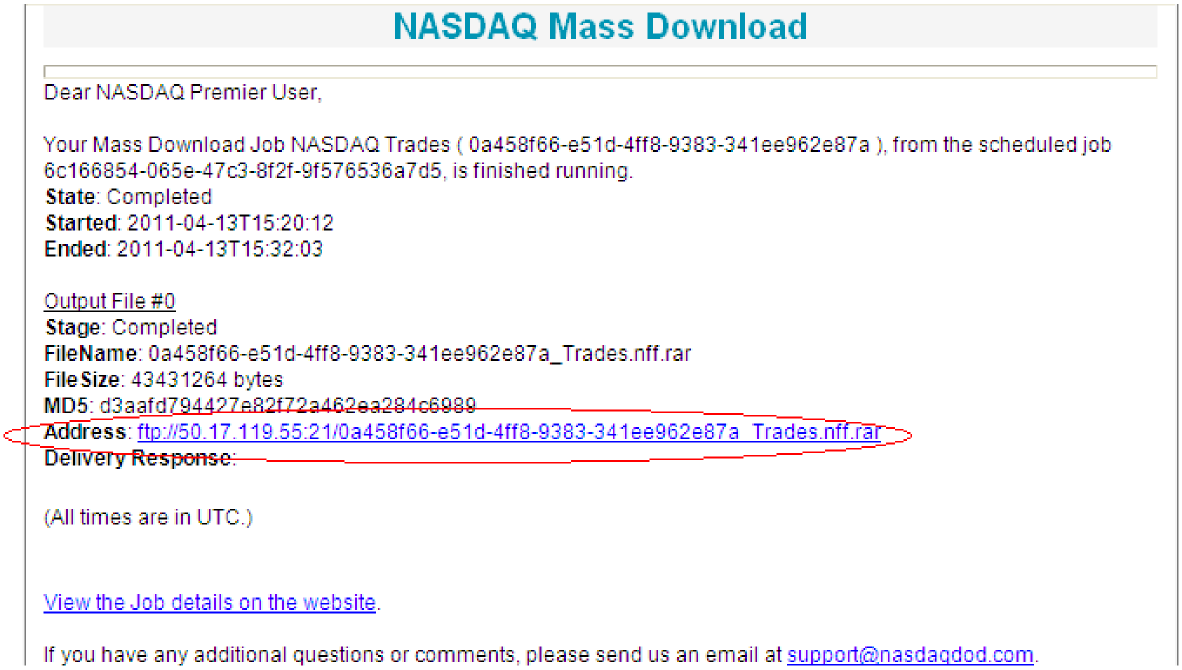
Change Your FTP Password

FTP Username: test

New FTP Password:

New FTP Password Confirm:

2. Enter your new password and select the 'Updated Password' button.
3. To start using Mass Downloads, go to the Mass Download Page and create a new Mass Download Job (one-time or recurring) by selecting the 'Create New Job' button.
4. After you submit a new job, you can continue to check progress on the Mass Download Page.
5. Once your job completes running, you will receive notification via email with a link to the location of your dataset.



NASDAQ Mass Download

Dear NASDAQ Premier User,

Your Mass Download Job NASDAQ Trades (0a458f66-e51d-4ff8-9383-341ee962e87a), from the scheduled job 6c166854-065e-47c3-8f2f-9f576536a7d5, is finished running.
State: Completed
Started: 2011-04-13T15:20:12
Ended: 2011-04-13T15:32:03

Output File #0
Stage: Completed
FileName: 0a458f66-e51d-4ff8-9383-341ee962e87a_Trades.nff.rar
File Size: 43431264 bytes
MD5: d3aafd794427e82f72a462ea284c6989
Address: ftp://50.17.119.55:21/0a458f66-e51d-4ff8-9383-341ee962e87a_Trades.nff.rar
Delivery Response:

(All times are in UTC.)

View the Job details on the website.

If you have any additional questions or comments, please send us an email at support@nasdaqdod.com.

6. Use your favorite FTP software to download your files from the URIs shown in the "Address" fields in your e-mail notification as shown above. If uncertain about what FTP software to use, we recommend FileZilla.
   a. Enter the Host and Port Number from the Address provided in the Job Completion email you received. For example, ftp://184.73.102.165:21/ba8c880b-1052-4d39-a56c-

2a7bbfece419_EndOfDayData.nff.rar. The Host is the highlighted green and the port is the highlighted yellow.

b. Enter your FTP Username/Password as defined in step 1.

c. Press Ok to download the dataset.

CAN I PROGRAM MY OWN SOFTWARE APPLICATION SO THAT IT CAN SYSTEMATICALLY CREATE, SCHEDULE AND MANAGE JOBS THAT GENERATE MASS DOWNLOAD DATA FILES?

Yes. You can use the NASDAQMassDownload web service as an API to let your own application perform all of the following functions systematically:

- Create and schedule jobs
- View the current status of your jobs
- Cancel jobs
- Redeliver files from completed jobs without having to rerun the jobs again

For more information, visit the NASDAQMassDownload web service page at http://www.nasdaqdod.com/NASDAQMassDownload.asmx.

AFTER I SUBMIT A REQUEST, HOW DO I RETRIEVE MY DATA?

You can retrieve data by downloading from an FTP site or requesting data be sent directly to an existing Amazon s3 cloud location.

WHEN MAKING REQUESTS THROUGH MASS DOWNLOADS USING A WEBSERVICE/API, ARE THERE ANY LANGUAGE RESTRICTIONS?

There are no language restrictions when making requests through Mass Downloads using a webservice/API. The web service is technology agnostic, just like the quotes, trades & analytics web services.

CAN I SCHEDULE AUTOMATIC DOWNLOADS?

Yes, you have the option to schedule automatic daily downloads. Simply choose that “Daily” option in the **Recurrence** field.

WILL I BE NOTIFIED ONCE MY JOB IN COMPLETE?

Yes. We will send you notification when your files are available for download through either Email, HTTPGet or HTTPPost. For more information please visit: http://www.nasdaqdod.com/NASDAQMassDownload.asmx.

DO YOU SUPPORT BOTH FTP AND FTTPS PROTOCOLS?

Yes, FTPS and FTP are both supported.

IN WHAT FORMAT ARE FILES DELIVERED?

You select the format of your file delivery using a field delimiter (comma or pipe delimited). To facilitate bandwidth, we recommend your firm download files in the csv format. In addition, csv formats have

- Smaller sizes
- Faster download times
- Ease of loading into a database tableSmaller storage requirements

WHEN I SUBMIT A REQUEST, WILL I RECEIVE ALL DATA IN ONE FILE?

It depends on the size of your data set. For small data sets, you will receive one file. For larger data sets, we separate the file in to smaller files. This is to allow earlier downloading of files. Instead of having to wait for the entire job to finish, you only have to wait for the first file to finish. Then you can start downloading in parallel while the remaining files continue to be created as part of the same job. This is beneficial for customers who have time sensitive requirements.

WHAT COMPRESSION OPTIONS ARE AVAILABLE WITH MASS DOWNLOAD FUNCTIONALITY?

NASDAQ Data-On-Demand offers the following compression types:

- Zip
- GZip
- RAR
- BZip2
- 7Zip

WHAT COMPRESSION TYPE IS THE BEST AND MOST EFFICIENT TO USE?

Any compression type is efficient, however we recommend using RAR.

WHAT HAPPENS IF THERE IS AN UNEXPECTED INTERRUPTION IN THE PROCESSING OF A JOB OR WHEN DOWNLOADING FILES?

If the job itself fails, the job must be rerun. If the job succeeded in creating the file, but the transfer to the FTP server or S3 bucket failed, then the job doesn't need to be rerun. Simply press the '**Redeliver**' link in the job queue and your data will be delivered much faster.

WHAT HAPPENS IF THERE IS AN UNEXPECTED INTERRUPTION WHEN I AM DOWNLOADING FILES?

Reinitiate download. DOD supports resuming downloads that were interrupted, therefore, the download will not start from the beginning again, but pick up where it left off.

WHY DOES DATA-ON-DEMAND ALLOW ONE SIDED QUOTES?

DOD functions as a pass through of the bid/ask information directly from the Security Information Processors (SIP). Therefore, users may see one sided quotes (0 priced asks or 0 priced bids) in the quotes data set. As using a 0 ask can appear to create a crossed market, customers should ignore the creation of a crossed market when creating the NBBO. Refer to the UTP Quotation Data Feed Specification and the Consolidated Quotation System Specification guidelines to create the NBBO or any other metrics based on quote data.

WILL I BE CHARGED FOR DOWNLOADING DATA MULTIPLE TIMES?

No, you can download the data query as many times as you like for internal distribution. You will only be charged for downloading the original bandwidth of data.

WHAT TYPE OF SECURITIES ARE AVAILABLE FOR DOWNLOAD?

You can download NASDAQ, NYSE, AMEX, OTCBB and OTC Market listed securities. You can download by listing market. If you only need a subset of securities, you can upload a csv file in to our system with the list of symbols you need.

WHAT DATA SETS CAN I DOWNLOAD UNDER THE MASS DOWNLOAD FUNCTIONALITY?

The Operations currently available with Mass Download capability are:

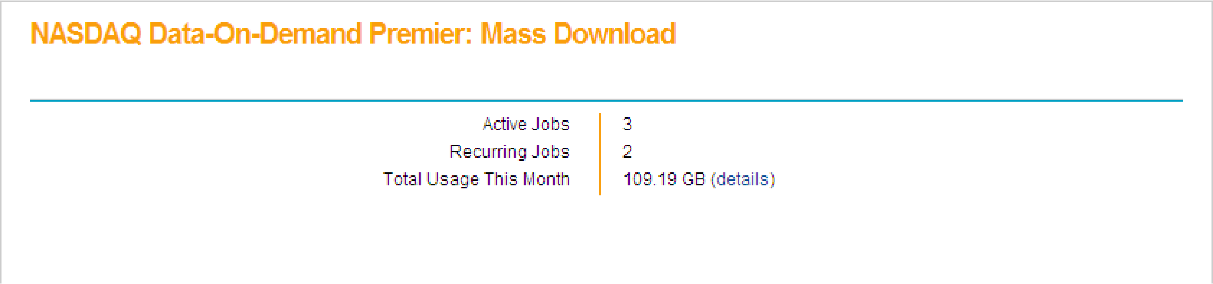
- GetTrades
- GetQutoes
- GetVWAP
- GetEndofDayData

Coming Soon:

- GetNBBO
- GetQuotesHighsAndLows
- GetSummarizedTrades
- SearchQuotesByBidAskPrice
- SearchQuotesByBidAskQuantity
- SearchTradesByPrice
- SearchTradesByQuantity
- SearchTradesBySaleConditions
- SearchQuotesBySpread
- GetAverageMarketSpread

WHERE CAN I VIEW MY TOTAL USAGE FOR THE MONTH?

To determine how much data you've downloaded, log in at nasdaqdod.com, go to **My Account** and select Mass Download. You'll see how much you've downloaded at the top of the Job Queue section:



NASDAQ Data-On-Demand Premier: Mass Download

Active Jobs	3
Recurring Jobs	2
Total Usage This Month	109.19 GB (details)

II. WEB SERVICES FUNCTIONALITY

WHERE CAN I FIND API DOCUMENTATION?

To access API documentations for the web service operations, please go to the individual operation page at www.nasdaqdod.com.

HOW DO I CHOOSE BETWEEN SOAP AND HTTP TO ACCESS OUR WEB SERVICES?

One the first and most important decisions you need to make when accessing our services is whether your will use the SOAP protocol or the HTTP protocol:

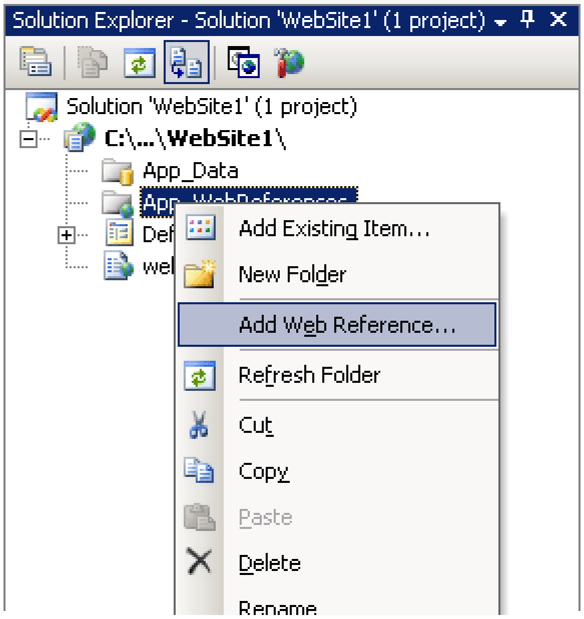
1. Using HTTP requires little in terms of technology but can involve more programming. All you need is to make sure that your development environment lets you pull data from a url. If you can do that, you can use our services. You will receive an XML document that you will need to parse. This requires an XML parser and can be more arduous.

2. Using SOAP requires that you have a SOAP toolkit but is much easier to implement. Many SOAP toolkits exist today in all environments. Most of them are free. Many environments also support SOAP natively (at least in their latest versions). A SOAP toolkit will be able to automatically 'discover' and 'understand' our web services. And instead of parsing XML documents, you will be working with classes (or objects, or complex types). This will make your work easier. This is why we recommend you use SOAP.

HOW DO I ADD A WEB REFERENCE OF THE WEB SERVICES IN VISUAL STUDIO 2005?

If you are using Microsoft Visual Studio as your development environment, the first thing you need to do to consume our services is to add a web reference. Microsoft has made this process incredibly easy. Just follow these simple steps and you will be up and going in 2 minutes:

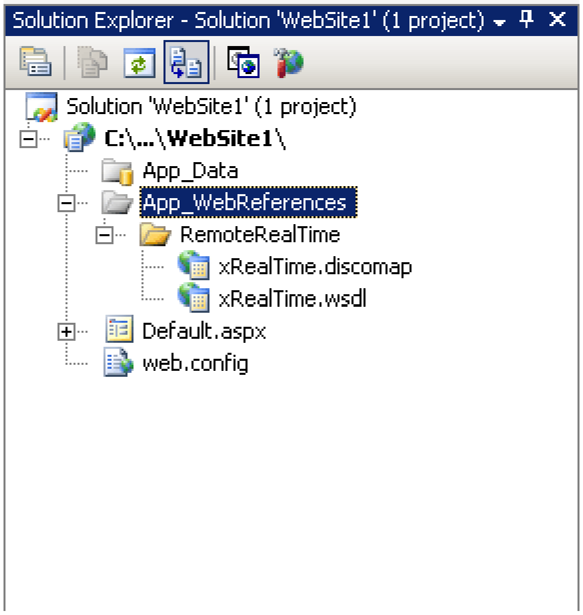
1. Open an existing project or create a new one. It does not matter what type of project it is (console, windows, web...)
2. Make sure the **Solution Explorer** window is open. If not, use the **View/Solution Explorer** menu to open it.
3. Locate the **Reference** entry in the Solution Explorer and right-click on it.
4. Choose **Add Web Reference** from the pop-up menu. This will open the **Add Web Reference** window.



5. Type in the address of the **WSDL** for the web service you want to use in the **Address** field.
6. You may find it easier to have a browser window open pointing to the WSDL for the service and cut and paste the address in the **Add Web Reference** window. The web references for all our web services use the same syntax: **http://ws.nasdaqdod.com/v1/{YourWebServiceName}.asmx?WSDL**. To find the WSDL of our web services, go to the web services product page. There are multiple ways to do that, such as through **Home Page**, **Main Menu -> Product Pricing** etc. When you are on our web service product page, there is a **WSDL** link on the left side. Click there to go to the WSDL page which shows you the path for that particular WSDL. Cut & Paste that value into the **Add Web Reference** window in Visual Studio
7. Click on the **Go** button on the right of the input field. Visual Studio will then display the WSDL, and activate the **Add Reference** button as shown below. If you get an error at this stage (e.g. *The HTML document does not contain Web service discovery information.*), you probably did not type in the right WSDL address.

8. You might want to change the name of the Web Reference. By default, Visual Studio will name it **com.domain name.www**. We do not find these default names too convenient. And we would recommend naming it to something else like **RemoteTrades**, or **RemoteQuotes** based on the name of the service.
9. Then click on the **Add Reference** button. Visual Studio will then close the window and dynamically generate a proxy class for the web service.

10. You will now see a new entry under **Web References** which is a link to that proxy class.

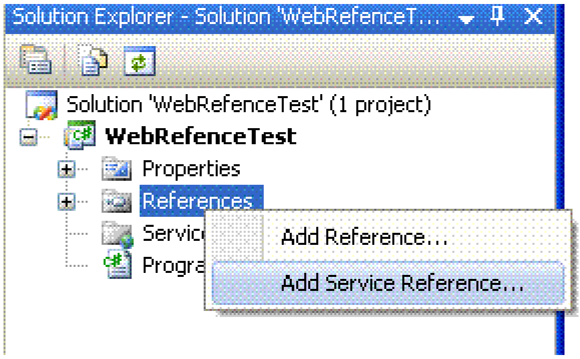


11. You are done! Your next step will be to write code to invoke the service. We actually provide sample code that you should be able to cut and paste and use right away. Simply go back to the Service page, choose an operation, and click on the Sample Code to check out VB or C# sample code for that operation. You should be able to cut and paste the code, compile and run it!

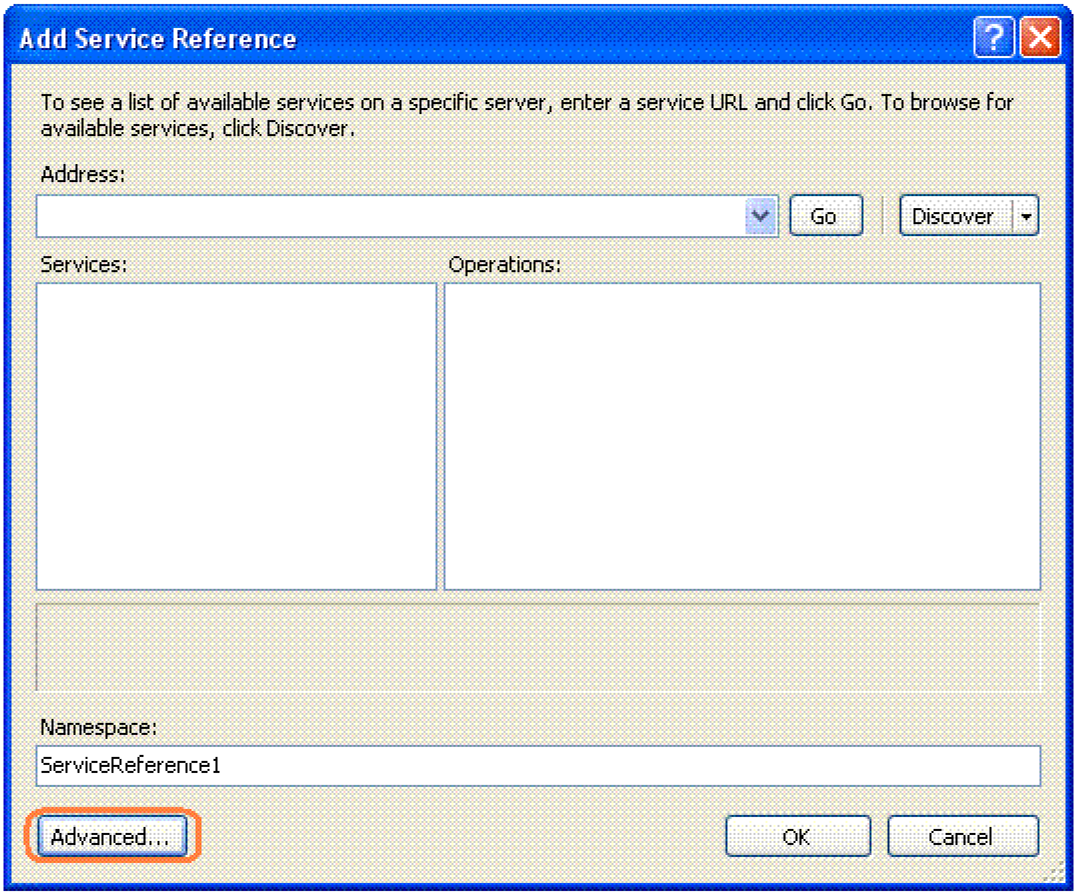
How Do I add a Web Reference of the Web Services in Visual Studio 2008?

If you are using Microsoft Visual Studio 2008 as your development environment, you'll just need to make a couple of extra clicks first, then follow the steps listed above under **How to add a Web Reference of the Domain name Web Services in Visual Studio 2005** starting from the **Type in the address of the WSDL for the web service you want to use in the Address field** section:

1. Open an existing project or create a new one. It does not matter what type of project it is (console, windows, web...)
2. Make sure the **Solution Explorer** window is open. If not, Use the **View/Solution Explorer** menu to open it.
3. Locate **Reference** entry in the Solution Explorer and right-click on it.
4. Choose **Add Service Reference** from the pop-up menu. This will open the **Add Web Reference** window.



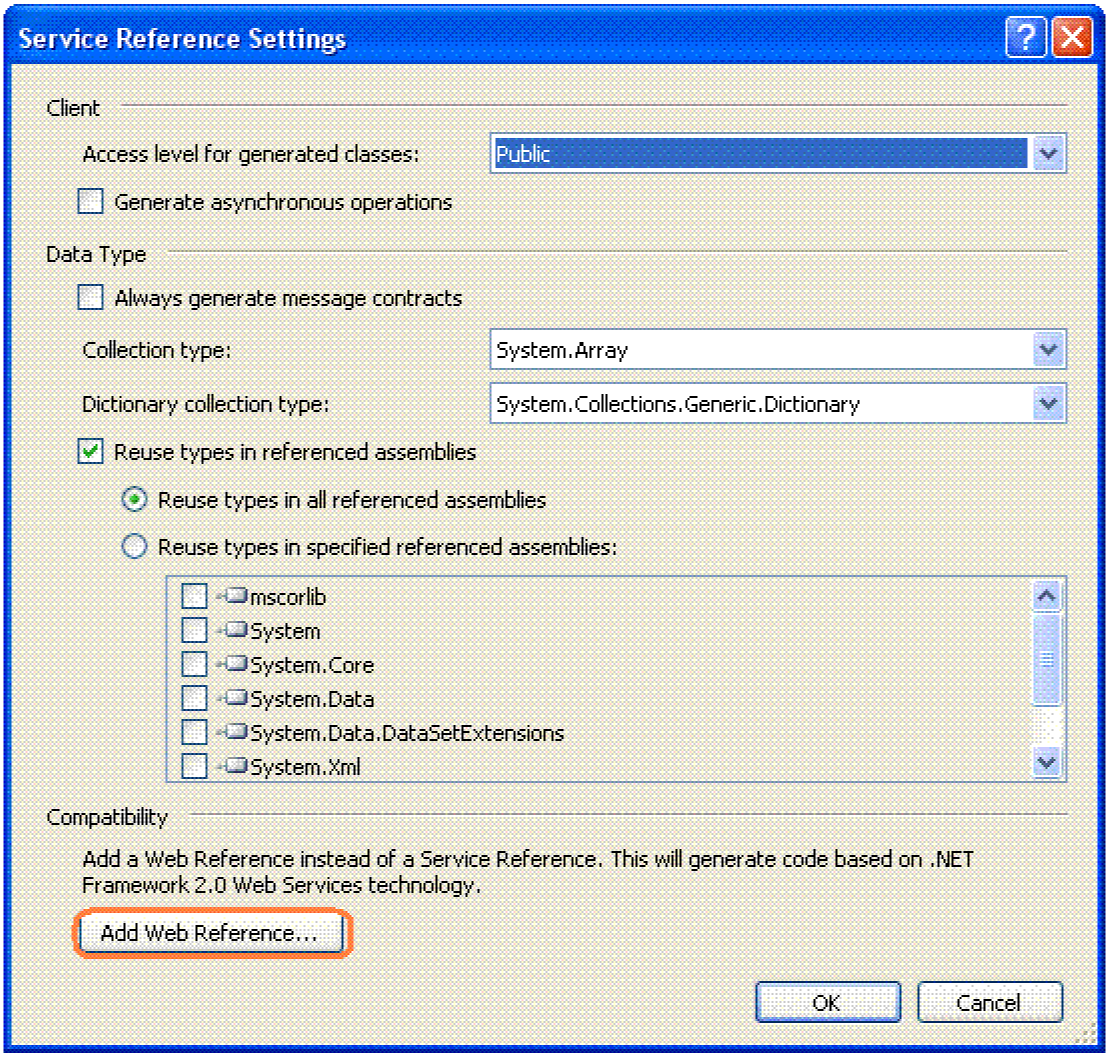
5. Click the **Advanced button** at the bottom left. This will open the **Service Reference Settings** window.


Add Service Reference

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

ServiceReference1

6. Click the **Add Web Reference button** at the bottom left. This will open the **Add Web Reference** window.



Service Reference Settings

Client

Access level for generated classes: Public

□ Generate asynchronous operations

Data Type

□ Always generate message contracts

Collection type: System.Array

Dictionary collection type: System.Collections.Generic.Dictionary

☑ Reuse types in referenced assemblies

● Reuse types in all referenced assemblies

○ Reuse types in specified referenced assemblies:

□ mscorlib
□ System
□ System.Core
□ System.Data
□ System.Data.DataSetExtensions
□ System.Xml

Compatibility

Add a Web Reference instead of a Service Reference. This will generate code based on .NET Framework 2.0 Web Services technology.

7. Follow the steps listed above *under* **How to add a Web Reference of the Web Services in VisualStudio 2005** *starting* from the **Type in the address of the WSDL for the web service you want to use in the Address field** section.

WHAT ARE THE DATA ELEMENTS RETURNED BY THE WEB SERVICES OPERATIONS?

Return classes for Domain name Web Services share a set of standard elements that can help you work more effectively. All return classes inherit from a class called **Common** which is defined in the WSDL of all services. This class includes the following elements:

Outcome

Is an enumeration indicating the outcome of the request. It will always be present and can therefore safely be tested. The possible values are:

1. **Success (0)**: The Request executed properly. Most return values should be populated.
2. **SystemError (1)**: An abnormal error occured when trying to execute this request, probably indicative of a bug with the service. Please contact our Support team if you run into such error. We track them on an ongoing basis and attempt to remove them as they occur.
3. **RequestError (2)**: Something was wrong with the request parameters, e.g. required parameters were missing or invalid, no data was found matching criteria...
4. **RegistrationError (3)**: You exceeded the number of requests allowed by your registration or subscription. See our **Registration Policy** section for details. Typical remedy include registering for the service if you have not done so or subscribing to the service.

Message

Is a string that contains an information message helping you further diagnose a problem indicated by **Outcome**. **Message** will is empty when **Outcome** is *Success*. Message will always contain some informational message if **Outcome** is not *Success*.

Identity

Is a string that indicates how your request was authenticated. This is helpful to diagnose registration problems. See the following section for details on authentication methods. Possible values are:

1. **IP**: The Request was authenticated using the requesting computer's IP address.**Cookie**: The Request was authenticated using an HTTP Cookie. (See SOAP Requests Using HTTP Cookies for details).
2. **Header**: The Request was authenticated using the SOAP Header.(See SOAP Requests Using SOAP Headers for details).
3. **Request**: The Request was authenticated using a Request Parameter. (See Working with Web Services as XML Document for details).

Delay

Is a string indicating how long it took our servers to process your requests (in seconds). Not included in this delay are network transition and system initialization times.

Dealing with Arrays

Return classes can consist of a single instance or an array (collection) of instances (indicated as ArrayOf... in the WSDL). When the return class is a single instance, you simply need to test for **Outcome, Message, Identity** or **Delay** in that instance. If the result is an array, you should always get at least one instance in the array. If an error occurred, it will be displayed in that first instance. It is therefore safe to test for the values above in the first instance of the return array.

Message and Outcome in SubClasses

Some web services provide message information in a subclass of the return class. For instance, if you use the GetQuotes operation, you will get an array of quotes as a result. If the system is unable to retrieve some, but not all quotes, the system will return a specific message for that quote instance.

DOES YOUR WEB SERVICE REQUEST REQUIRE AUTHENTICATION?

Yes, Domain name services and operations require authentication. Only Operations whose usage is Unlimited require no authentication. Domain name supports four methods to authenticate your requests:

IP Address

This method relies on the IP of the computer making the request. If your computer uses a **static IP**, it is the simplest approach. Domain name enables you to register an **unlimited** number of IPs. You should not use this method if you computer uses a **Dynamic IP address**. Your IP address is likely to be dynamic if:

1. You use a dial-up ISP service.
2. Your computer is not used as a server in your firm. You should check with your system administrator to be sure.

Soap Header

This is the best method to use when using SOAP.

This method relies on additional parameters passed along with your SOAP request to authenticate it. You should choose this method if your IP address is **Dynamic** and if you plan to use the **SOAP protocol**.

Check the SOAP Requests Using SOAP Headers tip for details.

HTTP Cookie

This method mainly applies to request you place using your browser. When you register for a service, we install a cookie on your machine. Subsequent requests you place use that cookie for authentication.

You can manually add or remove a cookie from your machine using the AddCookie and DeleteCookie operations.

You can programatically pass along with your GET or POST request to authenticate it. You should choose this method if your IP address is **Dynamic** and if you plan to use the **POST or GET protocols**. You can use this method if you use the SOAP protocol, but you will need to manually add the cookie to your request, check the SOAP Requests Using HTTP

Cookies section for details.
This is the best method to use when using HTTP.

Request Parameter This method relies on an additional parameter ("Header_Username" populated with the email address provided during registration) passed along with your GET or POST request to authenticate it. You should choose this method if your IP address is **Dynamic** and if you plan to use the **POST or GET protocols**. This method is most helpful if you like to manipulate web services as XML Document, check the Working with Web Services as XML Document section for details.

For HTTP requests, our web services try to authenticate your requests in the following order:

1. Request
2. Cookie
3. IP

For SOAP requests, our web services try to authenticate your requests in the following order:

1. SOAP
2. Request
3. Cookie
4. IP

If the system fails to authenticate your request, it will assume that you are **unregistered** for the web service and it will apply unregistered users restrictions. This will not necessary mean that your request will fail. Some operations allow a limited number of free requests to unregistered users.

How Do I authenticate SOAP Requests using SOAP Header?

If your computer uses a **Dynamic IP address** and you plan to use SOAP to invoke operation, you should add a SOAP header to your request to help authenticate it. You may also choose this approach to authenticate your requests even if your IP address is static. You only need to make sure you use the same email address when you registered.

All Domain name operations accept a SOAP Header simply named **Header.** This Header accepts three optional parameters.

1. **Username** holds your system **Username**, which is the **email address** you provided when registering for the service.
2. **Password** is currently unused.
3. **Tracer** is used for tracing your SOAP requests. See the **Troubleshooting** section for details.

To authenticate your request using SOAP headers, you need only provide a value for the **Username** parameter. The technique to add a soap header to your SOAP request will depend on your development toolkit.

How Do I authenticate SOAP Requests using Cookies?

While cookies work well with HTTP Get and POST requests, they will not automatically be passed when doing SOAP requests in most development toolkits. You should check with your development toolkit to ensure that the cookies are sent with your request. If you are using **Visual Studio.Net**, following are some instructions you can use to do this easily.

Adding HTTP Cookies to a SOAP Request in VisualStudio.Net

Simply add a routine to the web service proxy created by VS.Net and call that routine prior to invoking an operation.

1. Add a Web Reference to the web service you want to use by right-clicking on the **References** (or **Web References** if you have one) folder in the **Solution Explorer** window.
2. Enter the address of the web service WSDL you want to use, e.g. *http://ws.nasdaqdod.com/v1/NASDAQQuotes.asmx?WSDL*. Click on the **Add Reference** button.
3. Locate the web reference you created for the NASDAQQuotes web service in the **Solution Explorer** window.
4. Rename the Web Reference you just created by right-clicking on it and choosing **Rename** in the menu. For instance, you could use **RemoteQuotes** as the new name.
5. Using the Visual Studio.Net main menu, do "File", "Open", and "File". This will open an "Open File" window.
6. Click on the Web Reference directory. This is where VS.Net stores the proxy classes created automatically.
7. Click on the Folder for that Web Reference, e.g. **RemoteQuotes.**
8. In that folder you should find 3 files:
    1. a .map file
    2. a .wsdl file which is a local copy of the web service WSDL
    3. a .vb or .cs file which is the code generated by VS.Net for the proxy
9. Open the .vb or .cs file.
10. In the file, you will see the following code:

VB

```
Public Class RemoteQuotes
Inherits System.Web.Services.Protocols.SoapHttpClientProtocol

'<remarks/>
Public Sub New()
MyBase.New()
```

```
Me.Url = "http://ws.nasdaqdod.com/v1/NASDAQQuotes.asmx "
End Sub
```

C#

```
public class NASDAQQuotes : System.Web.Services.Protocols.SoapHttpClientProtocol {

///<remarks/>
public NASDAQQuotes() {
MyBase.New()
this.Url = "http://ws.nasdaqdod.com/v1/NASDAQQuotes.asmx";
}
```

11. Under that section, add a new routine called **AddCookie().** This routine creates a cookie and adds it to the cookie collection of the proxy class. Replace the **youremailaddress** value with your email address.

VB

```
Public Sub AddCookie()
Dim objCookie As New System.Net.Cookie("nasdaqemail","youremailaddress", "/",
"http://ws.nasdaqdod.com/v1/NASDAQQuotes.asmx")
Dim objCookieContainer As New System.Net.CookieContainer(1)
objCookieContainer.Add(objCookie)
Me.CookieContainer = objCookieContainer
End Sub
```

C#

```
public void AddCookie()
{
System.Net.Cookie objCookie = new System.Net.Cookie("nasdaqemail","youremailaddress",
"/", "http://ws.nasdaqdod.com/v1/NASDAQQuotes.asmx");
System.Net.CookieContainer objCookieContainer = new System.Net.CookieContainer(1);
objCookieContainer.Add(objCookie);
this.CookieContainer = objCookieContainer;
}
```

12. Then you code must call this method **prior to each time** a web service method is called. It is not sufficient to call **AddCookie()** once.

    VB

    ```
    Dim objRemoteQuote as New RemoteQuotes.NASDAQQuotes()
    'Add this prior to each method call
    objRemoteQuote.AddCookie()
    Dim objExtendedQuote as RemoteQuotes.ExtendedQuote =
    objRemoteQuote.GetQuote("MSFT")
    ```

    C#

    ```
    RemoteQuotes.NASDAQQuotes objRemoteQuote = new RemoteQuotes.NASDAQQuotes();
    ///Add this prior to each method call
    objRemoteQuote.AddCookie();
    RemoteQuotes.ExtendedQuote objExtendedQuote = objRemoteQuote.GetQuote("msft");
    ```

How Do I authenticate SOAP Requests using HTTP Cookies?

Doing POST and GET requests using HTTP cookies is pretty straightforward. When you register for a service or operation, the system will create a cookie on your computer. Any POST or GET request you place later on should automatically forward the cookie to the servers.

If you delete your cookie or want to add the cookie on another machine, you can easily recreate it by logging in to our website using the same email address you used to register.

How Do I call the web services using HTTP POST or GET?

Some developers prefer working with web services as XML Documents (HTTP) rather than instances of objects (SOAP). The advantage of such approach is that one can readily apply an XSL Stylesheet to the result of the request to reformat the output. This can be done directly in an ASPX page without much coding.

There are two things to consider when using this approach:

1. If you used a Dynamic IP registration, you should place your requests using the GET protocol and qualify them with the optional 'Header_Username' parameter populated with the email address you provided at registration. This is needed because SOAP Headers and Cookies will not work with this approach. If you are using Static IP registration, this step is not required. If you use POST, you must simply add the 'Header_Username=YourEmailAddress' to the

parameter pairs you post.

2. You should be aware that the resulting document is defined with "http://www.nasdaqdod.com/services" as its **default namespace**. Therefore all XML manipulations of that document will require that namespace to be defined and all queries to be prefixed accordingly.

   Following is sample code illustrating this method in Visual Studio.Net:

VB

```
'create an xml document instance
Dim objDocument as New System.Xml.XmlDocument()
'load the document by providing a url to the service fully
'qualified with all required parameters of the service
'if you use Dynamic Ip Registration, you must also provide the 'Header_Username'
'parameter populated with the email address you provided at registration
objDocument.Load("http://ws.nasdaqdod.com/v1/NASDAQQuotes.asmx/GetMarketSummary?_H
eader_Username=youremail@yourcompany.com")
'you can now manipulate the content of the document
'first create a namespace manager instance
Dim objNameSpaceManager As New
System.Xml.XmlNamespaceManager(objDocument.NameTable)
'then add the Nasdaq default namespace to it. The prefix name 'xi' is arbitrary
objNameSpaceManager.AddNamespace("xi",
"http://ws.nasdaqdod.com/v1/NASDAQQuotes.asmx/services/")
'any Xpath query must use the namespace manager
'all elements and attribute references must be prefixed
'the following query queries the 'Index' rows from the document.
Dim objNodeList As System.Xml.XmlNodeList = objDocument.DocumentElement.SelectNodes _
("descendant::xi:Index", objNameSpaceManager)
'we can now process the nodes
Dim objNode as System.Xml.XmlNode
For Each objNode In objNodeList
   'we qualify the query again
   Console.WriteLine(objNode.SelectSingleNode("child::xi:Name",
objNameSpaceManager).InnerText)
Next
```

How to force the calls to use HTTP Get instead of HTTP Post?

Some proxy servers block HTTPPOSTcalls. To generate the client side code to use HttpGet instead of HttpPost, one can use the wsdl.exe generation tool with the following command:

```
    wsdl /l:CS /protocol:HttpGet < URL to the WSDL >
```

The drawback of doing it this way is that a lot of helpful information will be lost, e.g., the inputs to services are always strings and not enums. There is also no built in Header and Username by generating the code this way, so the calls one make would need to be authenticated through IP for this to really work.

Besides using the wsdl.exe generation tool, another option is to just use the WebRequest class to fecth the service output and deal with parsing the XML by hand.

How Do I apply an XSL stylesheet to the result of a web service?

If you used the method described in the previous topic, a natural next step would be to apply an XSL stylesheet directly to the result of the web service. Once you have loaded the result of a web service as XML document (as described in the previous FAQ topic), applying a stylesheet is simple. You still have to worry about manipulating the document using the proper namespace. For instance, the following code will apply a stylesheet directly to a document as it is loaded in an XML WebControl for display in a web page.

VB

```
'create an xml document instance
Dim objDocument as New System.Xml.XmlDocument()
'load the document by providing a url to the service fully
'qualified with all required parameters of the service
'if you use Dynamic Ip Registration, you must also provide the 'Header_Username'
'parameter populated with the email address you provided at registration
objDocument.Load("http://ws.nasdaqdod.com/v1/NASDAQQuotes.asmx/GetMarketSummary?_Header_Username=youremail@yourcompany.com")
'you can now manipulate display the content of the document after applying a stylesheet
'first create a XSL transform instance
Dim objStylesheet As New System.Xml.Xsl.XslTransform()
'then load your stylesheet
objStylesheet.Load("c:/thelocationofyourstylesheet.xslt")
'let's create an XML webcontrol
Dim objXMLControl As New System.Web.UI.WebControls.Xml()
'let's apply the document and stylesheet to it
objXMLControl.Document = objDocument
objXMLControl.Transform = objStylesheet
'you could then add this control to your page.
```

Let's find out what the stylesheet looks like. Remember that the output of the web service will look something like this (here we use the GetMarketSummary operation as an example). To process this document properly, the stylesheet must:

1. Declare the *default namespace* of the document (**http://ws.nasdaqdod.com/v1/NASDAQQuotes.asmx/services/**)
2. Qualify all xPath query with this namespace

Click here to see a sample stylesheet and here to see the result of that stylesheet applied to the service.

How do I create a reference assembly to a web service when using ASPX?

If you are planning to use one of our web services in an ASP.Net application and if **you are not planning to use the code-behind technique**, you will need to create a reference assembly to the service. To do this, follow these simple steps:

1. First, generate the proxy source file using the wsdl.exe utility at a Visual Studio .Net Command Prompt (here we are using the NASDAQAnalytics service as an example)
   - C:\>wsdl.exe /language:VB http:://ws.nasdaqdod.com/v1/NASDAQQuotes.asmx?WSDL
2. This will generate a file called NASDAQAnalytics.vb (or .cs if you used the CS language option). You can then compile the VB file into an Assembly.
   - C:\>vbc.exe /t:library /r:System.dll /r:System.Web.dll /r:System.Web.Services.dll /r:System.Xml.dll NASDAQAnalytics.vb
3. This will generate a file called NASDAQAnalytics.dll that you can add to your VisualStudio project and must copy to your ASP.NET project /bin directory.
4. You can then consume the service using code similar to that provided under the **Sample Code** in the **This Operation** menu.

Why does Data-On-Demand allow one sided quotes?

DOD functions as a pass through of the bid/ask information directly from the Security Information Processors (SIP). Therefore, users may see one sided quotes (0 priced asks or 0 priced bids) in the quotes data set. As using a 0 ask can appear to create a crossed market, customers should ignore the creation of a crossed market when creating the NBBO. Refer to the UTP Quotation Data Feed Specification and the Consolidated Quotation System Specification guidelines to create the NBBO or any other metrics based on quote data.