IMS Web Service
Technical Guide

Document History

08/30/2010	James Hogan hogan.james@imstransport.com	Document created. Outlines basic requirements, functions and samples
02/01/2011	James Hogan hogan.james@imstransport.com	New functions added to support IRIS dispatches: loadShippingLines, loadClient2List, loadClientLocations, submitDispatchXML
12/15/2011	James Hogan hogan.james@imstransport.com	Documentation for FasTrak Move Status Notification Service added
1/11/2012	James Hogan hogan.james@imstransport.com	Added working/sample C# encrypt() function
5/16/2013	James Hogan hogan.james@imstransport.com	Added function getQuotesForCE to retrieve all quotes for the top 5 closest empties to the given inland location
06/24/14	James Hogan hogan.james@imstransport.com	Added function zipSearch to retrieve all valid zipcodes for the given inland city, state.

Table of Contents

Introduction to the IMS Web Service

The IMS Web Service is a SOAP based web service The Web Service Definition Language can be accessed at the URL: http://webservice.imstransport.com/imsservice.php?wsdl

W3C documentation for WSDL is available at: http://www.w3.org/TR/wsdl

Audience

This document is intended to be used as a reference by technical personnel familiar with, at least, the follow types of technology:

1) SOAP Web Services
2) XML
3) Client-Service models

The samples referenced in this document as well as the samples provided by IMS assume that the reader is familiar with at least one of the following programming languages:

1) PHP
2) VB.NET

The PHP sample should provide a general concept to those familiar with languages such as: PERL, Python, C or C++. The VB.NET sample should provide a general concept to those familiar with languages such as: C# or Java

Rate Terms and Conditions

All rates generated by the IMS Web Service are bound by the same Rate Terms and Conditions as those generated by the IMS website and are subject to change. The current terms and conditions follow:

Rate Terms and Conditions
1. Quotes are proprietary and may not be disclosed to other parties without written permission from IMS.
2. Shipments must conform to legal federal and state highway limitations.
3. IMS is not responsible for overweight containers.
4. All rail shipments are subject to handling railroads' rules circular.
5. All rates are subject to change.
6. Cargo insurance is limited to $100,000 US per shipment.
7. IMS does not handle residential household goods shipments.
8. Rate quotes are valid for the equipment size, type, cargo weight, and dimension (if applicable) as quoted.
9. Rates quotes do not include fuel surcharge (f/s).
10. IMS is not responsible for empty terminal location.
11. All invoices not paid within credit terms will accrue interest at the rate of 2% per month from the date of the invoice until paid in full.
12. Other accessorial charges, including without limitation: Storage; Diversion Charge; Scale Tickets; Stop Off etc. will be determined by IMS and are not included in the rate quote.
13. Rates are per container/truck load.
14. Rate quotes do not include chassis splits or flip charges.
15. Door service includes two free hours to load/unload, $65 US per hour will be charged thereafter excluding commercial/local zones and exhibition cargo.
16. Domestic rates include two free hours for load/unload, $95.00 per hour will be charged thereafter.
17. IMS does not handle any Class 1 hazardous cargo or radioactive materials
18. Subject to chassis rental fee of $18.00 per day if applicable.

Any questions regarding the rates generated by the IMS Web Service should be directed to the IMS Rate Department.

Support

Systems and development support is available for the IMS Web Service through the IMS MIS department. Issues regarding rate quotes or logistics issues such as move locations and availability should be directed to the IMS Rate Department or the IMS Operations Department.

Requirements To Use IMS Web Service

Access to the IMS Web Service must be granted to your organization by IMS before you can begin to use the service. The following information must be made available to you by IMS in order to connect to the service.

1) **Web service Public Key** (provided by IMS – to be sent with every WS request)
2) **Web service Private Key** (provided by IMS – to be used to encrypt a user's password and never sent alone in a WS request)
3) Web service must be activated by IMS

Web Service Public Key

The public key is a 32 character string provided by IMS that uniquely identifies the request as coming from your company. The public key should be sent in each web service request. Although this is a “public key” it should be kept confidential and only used when communicating with the IMS Web Service

The public key must be passed into any functions that require a parameter with the name: **clientkey**

Web Service Private Key

The private key is a 32 character string provided by IMS that is used to encrypt your user's password before being sent as part of a SOAP message. **This key should always be kept confidential and should never be transmitted alone or in plain text** to IMS or any other service. The private key encryption helps to keep your user's password secure in case of a DNS attack or your client otherwise connecting to a malicious host.

The private key must be used to encrypt a user's password as demonstrated in the below functions. If you are writing a client in another language, you will need to write this function to operate in the same manner such that the output of your function matches the output of these sample functions. Any other encryption method will not be able to be decrypted by the server and will result in a login error.

```php
function encrypt($string, $key) {
  $result = '';
  for($i=0; $i<strlen($string); $i++) {
    $char = substr($string, $i, 1);
    $keychar = substr($key, ($i % strlen($key))-1, 1);
    $char = (ord($char)+ord($keychar)) . "|";
    $result.=$char;
  }
  return $result;
}
```

Above - PHP sample

```vb
  Private Function encrypt(ByVal password As String, ByVal key As String) As String
    Dim result As String
    Dim thischar As String
    Dim keychar As String
    Dim startchar As Integer

    result = ""
    For i As Integer = 0 To password.Length - 1 Step 1
      thischar = password.Substring(i, 1)
      startchar = (i Mod key.Length) - 1
      If startchar < 0 Then
        startchar = key.Length + startchar
      End If
      keychar = key.Substring(startchar, 1)
      thischar = (Asc(thischar) + Asc(keychar)).ToString + "|"
      result += thischar
    Next

    Return result

  End Function
```

Above - VB.NET Sample

```
 public string encrypt(string password, string key) {
   string result;
   char thischar;
   char keychar;
   string newkey;
   int startchar;
   int i;
   result = "";
```

```
    for( i = 0; i <= password.Length - 1; i++){
        thischar = (char)password[i];
        startchar = (i % key.Length) - 1;
        if( startchar < 0){
            startchar = key.Length + startchar;
        }
        keychar = key[startchar];
        newkey = ((int)thischar + (int)keychar).ToString();
        newkey += "|";
        result += newkey;
    }
    return result;
  }
```

Above - C#

IMS Web Service – Available Functions

The IMS Web Service provides only the functions needed to retrieve quotes from the IMS Rate System. Additional functions may become available in the future. All functions of the IMS Web Service require that the client provide the public key as the **clientkey** parameter as well a valid username and password with access to the system. The permissions, in terms of access to buy rates and/or sell rates are determined by the authentication of the username being impersonated by the web service client.

Function: getQuote

This is the primary function to retrieve quotes from the IMS Webservice. The following table provides the possible values for the parameters of the function.

Parameter	Possible Values	Description
clientkey	String	Key Provided by IMS to authenticate a web service request This is your unique public key.
username	String	Valid username of a user this web service is impersonating. Permissions (Buy/Sell Rate Visibility) will be applied by username
password	Encrypted String	Password for the above user encrypted with the Private Key provided by IMS. Be sure to pass the user's plain text password through the encrypt() function before sending to the web service.
MoveTypeId	1	1 = International
Mode	ALL TO	ALL =All Modes (Truck/Rail/Intermodal) TO = Truck Only
Service	D ND	D = Door ND = Non-Door
LiveDropInd	D L	D = Drop L = Live
LoadedEmptyInd	L E	L = Loaded E = Empty * - All Door moves are Loaded
ImportExport	I X B	I = Import (Valid with Service D - Door) X = Export (Valid with Service D - Door) B = Directional (Valid with Service ND - NonDoor)
ContainerSize	ALL 20 40/45	ALL / All Sizes (20/40/45) 20 / 20' Container only 40/45 / 40' and 45'containers
ContainerType	DV/HC DV FL HC OT RF RH	DV/HC = Dry Van/High Cube (standard equipment) DV – Dry Van FL – Flat Rack HC – High Cube OT – Open Top RF – Refrigerated RH – Refrigerated High Cube
P1	String	Valid IMS Fixed Location – See page 15 for Function: locEmpties()
P2	String	When Service = D Then Valid US ZipCode When Service = ND Then Valid IMS Fixed Location – See page 15 for Function: locEmpties()
P3	String	When Service = D Then Valid IMS Fixed Location – See page 15 for Function: locEmpties() When Service = ND Then Empty String – any values are ignored

Parameter	Possible Values	Description
Client2	String	Valid Company Client Name. The list of valid Company Client Names can be determined from the IMS Web App → Administration → Agent Mgt and provide the text of the Agent Company name.
HazardousInd	Y N	Y – Yes N - No
ReeferInd	Y N	Y – Yes N - No
OverweightInd	Y N	Y – Yes N - No

The getQuote function will return an XML string with the quote information. An XSD for the returned XML is below:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="data">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="result" type="xs:string" />
        <xs:element name="resultDesc" type="xs:string" />
        <xs:element name="search">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="mt" type="xs:unsignedByte" />
              <xs:element name="mode" type="xs:string" />
              <xs:element name="service" type="xs:string" />
              <xs:element name="livedrop" type="xs:string" />
              <xs:element name="loadedempty" type="xs:string" />
              <xs:element name="ix" type="xs:string" />
              <xs:element name="type" type="xs:string" />
              <xs:element name="size" type="xs:string" />
              <xs:element name="p1" type="xs:unsignedInt" />
              <xs:element name="p2" type="xs:unsignedInt" />
              <xs:element name="p3" />
              <xs:element name="client2" />
              <xs:element name="haz" type="xs:string" />
              <xs:element name="reef" type="xs:string" />
              <xs:element name="ow" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="quotes">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="quote">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="uid" type="xs:string" />
                    <xs:element name="originId" type="xs:string" />
                    <xs:element name="originName" type="xs:string" />
                    <xs:element name="destId" type="xs:string" />
                    <xs:element name="destName" type="xs:string" />
                    <xs:element name="emptyId" type="xs:string" />
                    <xs:element name="emptyName" type="xs:string" />
                    <xs:element name="importExportInd" type="xs:string" />
                    <xs:element name="containerSizeId" type="xs:string" />
                    <xs:element name="containerSizeDesc" type="xs:string" />
                    <xs:element name="containerTypeId" type="xs:string" />
                    <xs:element name="containerTypeDesc" type="xs:string" />
                    <xs:element name="mode" type="xs:string" />
                    <xs:element name="viaId" type="xs:string" />
                    <xs:element name="viaDesc" type="xs:string" />
                    <xs:element name="quoteNbr" type="xs:string" />
                    <xs:element name="hazardousFees" type="xs:string" />
                    <xs:element name="reeferFees" type="xs:string" />
                    <xs:element name="overweightFees" type="xs:string" />
                    <xs:element name="cleanTruckFees" type="xs:string" />
                    <xs:element name="quoteAmt" type="xs:string" />
```

```
                <xs:element name="fuelPct" type="xs:string" />
                <xs:element name="fuelFees" type="xs:string" />
                <xs:element name="basePlusSpecial" type="xs:string" />
                <xs:element name="allInRate" type="xs:string" />
                <xs:element name="quote2Amt" type="xs:string" />
                <xs:element name="fuel2Pct" type="xs:string" />
                <xs:element name="fuel2Fees" type="xs:string" />
                <xs:element name="base2PlusSpecial" type="xs:string" />
                <xs:element name="allIn2Rate" type="xs:string" />
                <xs:element name="effDate" type="xs:string" />
                <xs:element name="expDate" type="xs:string" />
                <xs:element name="createdBy" type="xs:string" />
                <xs:element name="createdOn" type="xs:string" />
                <xs:element name="requestedBy" type="xs:string" />
                <xs:element name="requestedByEmail" type="xs:string" />
                <xs:element name="requestedByPhone" type="xs:string" />
                <xs:element name="requestedByExt" type="xs:string" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

The following table details the return elements:

Element/Node	Possible Values	Description
data	--	Root Node – occurrs only once
data\result	ok notice error	ok – Indicates that the request was successful and quotes have been returned notice – Indicates that the request was successful but no quotes have been returned error – Indicates that the request was unsuccessful due to some parameters or system error
data\resultDesc	String	When result = ok Then this field will indicate the number of quotes that are being returned When result = notice Then this field will indicate "Rate To Follow". This is likely the result of the IMS Instant Rate Inquiry System (IRIS) being unable to generate a quote at this time. When result = error Then this field will indicate a description of the cause of the error. The possible list of error codes and their causes and expected resolutions will be provided later.
data\search	--	Node that container the values passed as input to the IRIS system. These should echo your original request parameters
data\search\mt	1	1 – Move Type International
data\search\mode	ALL TO	ALL - All Modes TO – Truck Only
data\search\service	D ND	D – Door ND - Non-Door
data\service\livedrop	L D	L – Live D - Drop
data\service\loadedempty	L E	L – Loaded E - Empty
data\service\ix	I X B	I – Import X – Export B - Directional
data\service\type	DV/HC DV FL HC OT RF RH	DV/HC = Dry Van/High Cube (standard equipment) DV – Dry Van FL – Flat Rack HC – High Cube OT – Open Top RF – Refrigerated RH – Refrigerated High Cube
data\service\size	ALL 20 40/45	ALL / All Sizes (20/40/45) 20 / 20' Container only 40/45 / 40' and 45'containers
data\service\p1	Int	IMS Location ID
data\service\p2	Int	IMS Location ID
data\service\p3	Int	IMS Location ID
data\service\client2	Int	IMS Client2 ID

Element/Node	Possible Values	Description
data\service\haz	Y N	Y- Yes N - No
data\service\reef	Y N	Y- Yes N - No
data\service\ow	Y N	Y- Yes N - No
data\quotes	–	Node that contains any number of quote node results – repeats as many times as the number returned in the data\resultDesc element.
data\quotes\quote	–	Node for a single quote
data\quotes\quote\uid	String	Unique ID for IMS system. Can be ignored
data\quotes\quote\originId	Int	IMS LocationId
data\quotes\quote\originName	String	Text description of the IMS Location
data\quotes\quote\destId	Int	IMS LocationId
data\quotes\quote\destName	String	Text description of the IMS Location
data\quotes\quote\emptyId	Int	IMS LocationId
data\quotes\quote\emptyName	String	Text description of the IMS Location
data\quotes\quote\importExportInd	I X B	I – Import X – Export B - directional/both
data\quotes\quote\containerSizeId	Int	IMS Container Size Id
data\quotes\quote\containerSizeDesc	20 40/45	Container Size Description
data\quotes\quote\containerTypeId	Int	IMS Container Type Id
data\quotes\quote\containerTypeDesc	DV/HC DV FL HC OT RF RH	IMS Container Type Id
data\quotes\quote\mode	R T I	R - Rail T – Truck I – Intermodal (truck and rail combination)
data\quotes\quote\viaId	Int	IMS Rail Service ID
data\quotes\quote\viaDesc	String	Rail Service Name if move to be completed by rail or intermodal service
data\quotes\quote\quoteNbr	String	IMS Quote Number
data\quotes\quote\hazardousFees	String	Added fee if the move includes hazardous changes.
data\quotes\quote\reeferFees	String	Added fee if the move includes reefer changes.
data\quotes\quote\overweightFees	String	Added fee if the move includes overweight changes.
data\quotes\quote\cleanTruckFees	String	Added fee if the move includes Clean Truck changes.

Element/Node	Possible Values	Description
data\quotes\quote\quoteAmt	String	Base quote amount
data\quotes\quote\fuelPct	String	Fuel Surcharge as a percentage
data\quotes\quote\fuelFees	String	Fuel cost (baseQuote * fuelPct)
data\quotes\quote\basePlusSpecial	String	Subtotal of baseQuote + fees (fuel not included)
data\quotes\quote\allInRate	String	Total quote amount baseQuote + fees + fuel
data\quotes\quote\quote2Amt	String	If the username has permission to view Sell Rates, then this is the baseQuote amount for the "Sell Rate"
data\quotes\quote\fuel2Pct	String	If the username has permission to view Sell Rates, then this is the fuel surcharge percentage for the "Sell Rate"
data\quotes\quote\fuel2Fees	String	If the username has permission to view Sell Rates, then this is the quote2Amt * fuel2Pct amount for the "Sell Rate"
data\quotes\quote\base2PlusSpecial	String	If the username has permission to view Sell Rates, then this is the subtotal of quote2Amt + fees (fuel not included)
data\quotes\quote\allIn2Rate	String	If the username has permission to view Sell Rates, then this is the total amount (quote2Amt + fees + fuel2Fees) for the "Sell Rate"
data\quotes\quote\effDate	String	Date that this quote is effective
data\quotes\quote\expDate	String	Data that this quote expires and will no longer be honored
data\quotes\quote\createdBy	String	Name of the user who created the quote
data\quotes\quote\createdOn	String	Date that the quote was originally created
data\quotes\quote\requestedBy	String	Name of the user who requested the quote (based on the username passed into the request)
data\quotes\quote\requestedByEmail	String	Email address of the user who requested the quote (based on the username passed into the request)
data\quotes\quote\requestedByPhone	String	Phone number of the user who requested the quote (based on the username passed into the request)
data\quotes\quote\requestedByExt	String	Phone extension of the user who requested the quote (based on the username passed into the request)

Function: getQuotesForCE

This function retrieves all quotes for the top 5 closest empty locations from the given inland location (P2) from the IMS Webservice. This function:

- **returns ONLY round-trip, door quotes.**
- **has the same output as the getQuotes function.**
- **may return more than 5 results if the IMS fixed location contains multiple service areas (example, quotes to Chicago, IL may return up to 3 quotes)**

The following table provides the possible values for the parameters of the function.

Parameter	Possible Values	Description
clientkey	String	Key Provided by IMS to authenticate a web service request This is your unique public key.
username	String	Valid username of a user this web service is impersonating. Permissions (Buy/Sell Rate Visibility) will be applied by username
password	Encrypted String	Password for the above user encrypted with the Private Key provided by IMS. Be sure to pass the user's plain text password through the encrypt() function before sending to the web service.
MoveTypeId	1	1 = International
Mode	ALL TO	ALL =All Modes (Truck/Rail/Intermodal) TO = Truck Only
LiveDropInd	D L	D = Drop L = Live
LoadedEmptyInd	L E	L = Loaded E = Empty * - All Door moves are Loaded
ImportExport	I X	I = Import (Valid with Service D - Door) X = Export (Valid with Service D - Door)
ContainerSize	ALL 20 40/45	ALL / All Sizes (20/40/45) 20 / 20' Container only 40/45 / 40' and 45'containers
ContainerType	DV/HC DV FL HC OT RF RH	DV/HC = Dry Van/High Cube (standard equipment) DV – Dry Van FL – Flat Rack HC – High Cube OT – Open Top RF – Refrigerated RH – Refrigerated High Cube
P2	String	When Service = D Then Valid US ZipCode When Service = ND Then Valid IMS Fixed Location – See page 15 for Function: locEmpties()
Client2	String	Valid Company Client Name. The list of valid Company Client Names can be determined from the IMS Web App → Administration → Agent Mgt and provide the text of the Agent Company name.
HazardousInd	Y N	Y – Yes N - No
ReeferInd	Y N	Y – Yes N - No
OverweightInd	Y	Y – Yes

Parameter	Possible Values	Description
	N	N - No

Function: locEmpties()

This function provides a necessary service that allows the developer to find the nearest IMS locations to originate or terminate a move. This function provides the interface to mimic the below on-screen functionality:

For example, if the user has an In-Land location of Eatontown, NJ (07724), you can pass the US ZipCode 07724 to this function and the function will return the list of "Closest Empty Locations". In this screen-shot, the locations highlighted in green are "common" locations.

Parameter	Possible Values	Description
clientkey	String	Key Provided by IMS to authenticate a web service request
username	String	Valid username of a user this webservice is impersonating. Permissions (Buy/Sell Rate Visibility) will be applied by username
password	Encrypted String	Password for the above user encrypted with the Private Key provided by IMS
dynLoc	Number – ZipCode or String – City,State	Provide a valid US ZipCode or a City, State for example: 07724 or Eatontown, NJ

The output XML from the locEmpties function will validate against the below XSD:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="data">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="result" type="xs:string" />
        <xs:element name="resultDesc" type="xs:string" />
        <xs:element name="locations">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="location">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="locationname" type="xs:string" />
                    <xs:element name="commonInd" type="xs:string" />
                    <xs:element name="miles" type="xs:string" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The following table explains the nodes in the XML document:

Element/Node	Possible Values	Description
data	--	Root Node – occurs only once
data\result	ok error	ok – Indicates that the request was successful and locations have been returned error – Indicates that the request was unsuccessful due to some parameters or system error
data\resultDesc	String	When result = ok Then this field will indicate the number of locations that are being returned When result = error Then this field will indicate a description of the cause of the error. The possible list of error codes and their causes and expected resolutions will be provided later.
data\locations	–	Node contains each of the location nodes that are a closest empty match – occurs as many times as the value of data\resultDesc
data\locations\location	–	Node containing location detail information.
data\locations\location\locationname	String	IMS Location name
data\locations\location\commonInd	Y N	Y – Location is a common location N – Location is not frequently used for moves
data\locations\location\miles	Int	Distance from the inland location to the IMS Closest Empty location.

Function: locSearch

This function provides a necessary service that allows the developer to find the names of IMS locations based on user input. The function provides the interface necessary to mimic the feature demonstrated in the below screen-shot.

For example, if the user is looking to move a container into the “North East Coast”, providing the text “north” will return the list of IMS Locations that begin with the matching text.

Parameter	Possible Values	Description
clientkey	String	Key Provided by IMS to authenticate a web service request
username	String	Valid username of a user this webservice is impersonating. Permissions (Buy/Sell Rate Visibility) will be applied by username
password	Encrypted String	Password for the above user encrypted with the Private Key provided by IMS
locName	String	The database is searched for locations beginning with this text

The output XML from the locEmpties function will validate against the below XSD:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="data">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="result" type="xs:string" />
        <xs:element name="resultDesc" type="xs:unsignedByte" />
        <xs:element name="locations">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="location">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="locationname" type="xs:string" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The following table explains the nodes in the XML document:

Element/Node	Possible Values	Description
data	--	Root Node
data\result	ok error	ok – Indicates that the request was successful and locations have been returned error – Indicates that the request was unsuccessful due to some parameters or system error
data\resultDesc	String	When result = ok Then this field will indicate the number of locations that are being returned When result = error Then this field will indicate a description of the cause of the error. The possible list of error codes and their causes and expected resolutions will be provided later.
data\locations	–	Node contains each of the location nodes that are a match
data\locations\location	–	Node containing location detail information.
data\locations\location\locationname	String	IMS Location name

Function: zipSearch

This function was added to provide the developer with the ability to retrieve the list of US Zip Codes that are valid for a given City/State combination.

The below table provides the input parameters for the inlandSearch() function.

Parameter	Possible Values	Description
clientkey	String	Key Provided by IMS to authenticate a web service request
username	String	Valid username of a user this webservice is impersonating. Permissions (Buy/Sell Rate Visibility) will be applied by username
password	Encrypted String	Password for the above user encrypted with the Private Key provided by IMS
city	String	The name of a US City.
stateabv	String	The two letter abbreviation of a US state.

The output XML from the zipSearch function will validate against the below XSD:

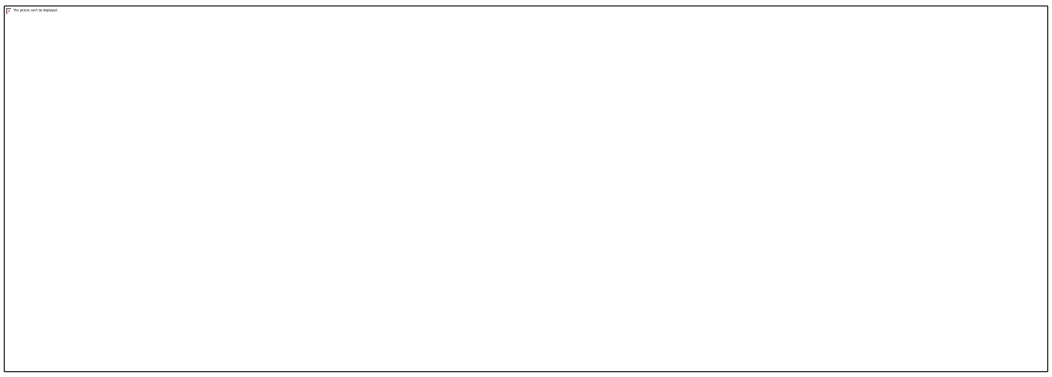
```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="data">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="result" type="xs:string" />
        <xs:element name="resultDesc" type="xs:unsignedByte" />
        <xs:element name="zips">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="zip">
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The following table explains the nodes in the XML document:

Element/Node	Possible Values	Description
data	--	Root Node
data\result	ok error	ok – Indicates that the request was successful and locations have been returned error – Indicates that the request was unsuccessful due to some parameters or system error
data\resultDesc	String	When result = ok Then this field will indicate the number of locations that are being returned When result = error Then this field will indicate a description of the cause of the error. The possible list of error codes and their causes and expected resolutions will be provided later.
data\zips	–	Node contains each of the zip nodes that are a match
data\zips\zip	–	Node containing IS Zip Code

Function: inlandSearch

This function provides a necessary service that allows the developer to find the names of inland locations, North American city names, based on user input. The function provides the interface necessary to mimic the feature demonstrated in the below screen-shot.



For example, if the user is looking to move a container from "Eatontown, NJ", providing the zipcode “07724” will return the list of cities that have the assigned zipcode. The user can also search by city name, for example, search for "Eaton" and see the cities that start with that name as below.

The below table provides the input parameters for the inlandSearch() function.

Parameter	Possible Values	Description
clientkey	String	Key Provided by IMS to authenticate a web service request
username	String	Valid username of a user this webservice is impersonating. Permissions (Buy/Sell Rate Visibility) will be applied by username
password	Encrypted String	Password for the above user encrypted with the Private Key provided by IMS
searchText	String	The database is searched for locations beginning with this text. The search text can be a city name or a us zipcode.

The output XML from the inlandSearch function will validate against the below XSD:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="data">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="result" type="xs:string" />
        <xs:element name="resultDesc" type="xs:unsignedByte" />
        <xs:element name="locations">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="location">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="locationname" type="xs:string" />
                    <xs:element name="imscity" type="xs:string" />
                    <xs:element name="imsstate" type="xs:string" />
                    <xs:element name="zipcode" type="xs:string" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The following table explains the nodes in the XML document:

Element/Node	Possible Values	Description
data	--	Root Node
data\result	ok error	ok – Indicates that the request was successful and locations have been returned error – Indicates that the request was unsuccessful due to some parameters or system error
data\resultDesc	String	When result = ok Then this field will indicate the number of locations that are being returned When result = error Then this field will indicate a description of the cause of the error. The possible list of error codes and their causes and expected resolutions will be provided later.
data\locations	–	Node contains each of the location nodes that are a match
data\locations\location	–	Node containing location detail information.
data\locations\location\locationname	String	Common Location name, state
data\locations\location\imscity	String	US Postal Service Location Name
data\locations\location\imsstate	String	US Postal Service Location State
data\locations\location\zipcode	String	US Postal Service Zip Code

Function: locGroupMembers

This function provides a necessary service that allows the developer to find the names of IMS locations that are members of a larger group, for example, the developer can find the names of the ports or ramps that make up the "NY/NJ (New Jersey)" group. The function provides the interface necessary to mimic the feature demonstrated in the below screen-shot.

IMS Rate Quotes for a Group imply that the rate is valid for any of the group members.

Parameter	Possible Values	Description
clientkey	String	Key Provided by IMS to authenticate a web service request
username	String	Valid username of a user this webservice is impersonating. Permissions (Buy/Sell Rate Visibility) will be applied by username
password	Encrypted String	Password for the above user encrypted with the Private Key provided by IMS
locName	String	Parent location name.

The output XML from the locGroupMembers function will validate against the below XSD:

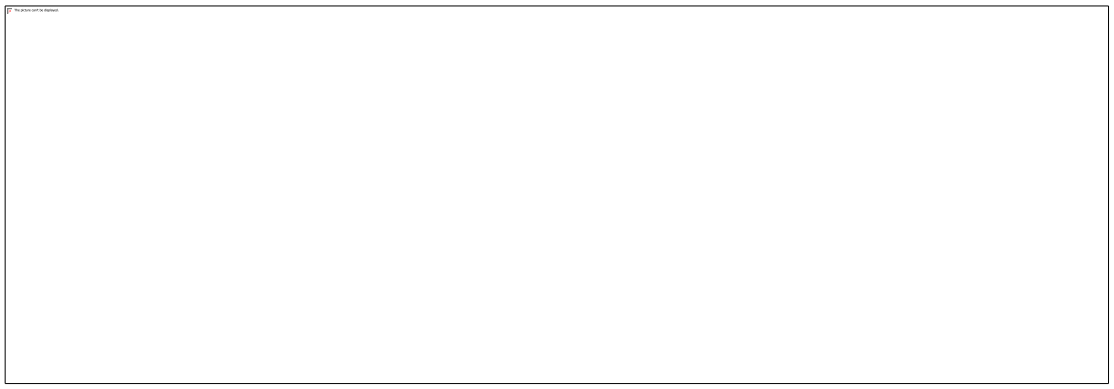
```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="data">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="result" type="xs:string" />
        <xs:element name="resultDesc" type="xs:string" />
        <xs:element name="locations">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="location">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="locationname" type="xs:string" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The following table explains the nodes in the XML document:

Element/Node	Possible Values	Description
data	--	Root Node
data\result	ok error	ok – Indicates that the request was successful and locations have been returned error – Indicates that the request was unsuccessful due to some parameters or system error
data\resultDesc	String	When result = ok Then this field will indicate the number of locations that are being returned When result = error Then this field will indicate a description of the cause of the error. The possible list of error codes and their causes and expected resolutions will be provided later.
data\locations	–	Node contains each of the location nodes that are a match
data\locations\location	–	Node containing location detail information.
data\locations\location\locationname	String	IMS Location name

Function: loadShippingLines

This function provides a necessary service that allows the developer to find the names of steam ship lines available for dispatches. The function provides the interface necessary to mimic the feature demonstrated in the below screen-shot.



Parameter	Possible Values	Description
clientkey	String	Key Provided by IMS to authenticate a web service request
username	String	Valid username of a user this webservice is impersonating. Permissions (Buy/Sell Rate Visibility) will be applied by username
password	Encrypted String	Password for the above user encrypted with the Private Key provided by IMS
shippingLine	String	Name of the shipping line you are searching for in the list. Provide an empty string to retrieve the full list.

The output XML from the loadShippingLines function will validate against the below XSD:

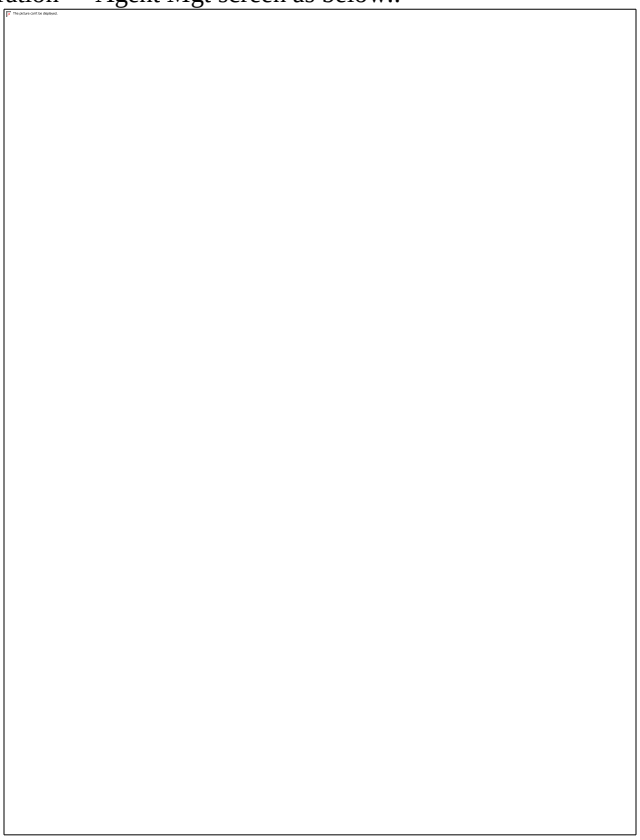
```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="data">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="result" type="xs:string" />
        <xs:element name="resultDesc" type="xs:string" />
        <xs:element name="shippinglines">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="shippingline">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="shippinglinename" type="xs:string" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The following table explains the nodes in the XML document:

Element/Node	Possible Values	Description
data	--	Root Node
data\result	ok error	ok – Indicates that the request was successful and shipping lines have been returned error – Indicates that the request was unsuccessful due to some parameters or system error
data\resultDesc	String	When result = ok Then this field will indicate the number of shipping lines that are being returned When result = error Then this field will indicate a description of the cause of the error. The possible list of error codes and their causes and expected resolutions will be provided later.
data\shippinglines	–	Node contains each of the shippingline nodes that are a match
data\shippinglines\shippingline	–	Node containing shipping line detail information.
data\shippinglines\shippingline\shippinglinename	String	Shipping Line Name

Function: loadClient2List

This function provides a necessary service that allows the developer to find the details of clients (Agents) available for quotes (Sell Rate, if available to your company) and dispatches. The function provides the interface to obtain the list of clients added from the Administration -> Agent Mgt screen as below:.



This function allows you to determine the possible list of clients for the "sell rate" in IRIS or for dispatches.

The names and all details for the Client list are entered through the IMS website if your company has access to the IMS "Sell System" and you have access to the "Agent Mgt" system. Current, in order to add Agents, you must manually enter them through the website by loggin to IMS website and going to the tab for Administration -> Agent Mgt If you do not see this tab, you may not have access to the IMS Sell System.

Parameter	Possible Values	Description
clientkey	String	Key Provided by IMS to authenticate a web service request
username	String	Valid username of a user this webservice is impersonating. Permissions (Buy/Sell Rate Visibility) will be applied by username
password	Encrypted String	Password for the above user encrypted with the Private Key provided by

		IMS
clientName	String	Name of the client (Agent) you are searching for in the list. Provide an empty string to retrieve the full list.

The output XML from the loadClient2List function will validate against the below XSD:

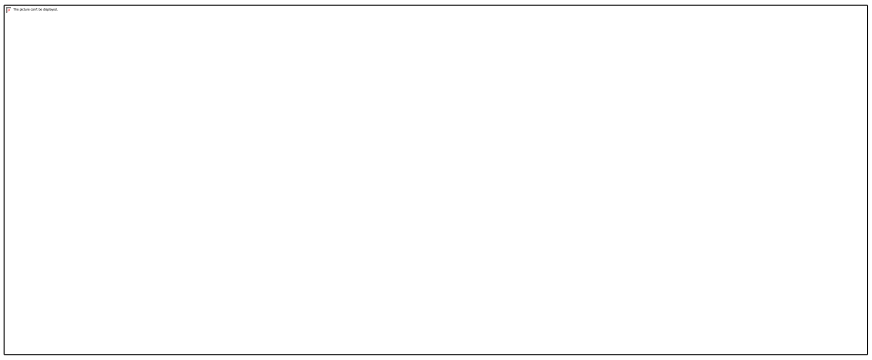
```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="data">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="result" type="xs:string" />
        <xs:element name="resultDesc" type="xs:string" />
        <xs:element name="clients">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="client">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ClientCompanyName" type="xs:string" />
                    <xs:element name="address1" type="xs:string" />
                    <xs:element name="address2" type="xs:string" />
                    <xs:element name="city" type="xs:string" />
                    <xs:element name="state" type="xs:string" />
                    <xs:element name="zipcode" type="xs:string" />
                    <xs:element name="country" type="xs:string" />
                    <xs:element name="telephone" type="xs:string" />
                    <xs:element name="fax" type="xs:string" />
                    <xs:element name="email" type="xs:string" />
                    <xs:element name="webpage" type="xs:string" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The following table explains the nodes in the XML document:

Element/Node	Possible Values	Description
data	--	Root Node
data\result	ok error	ok – Indicates that the request was successful and clients have been returned error – Indicates that the request was unsuccessful due to some parameters or system error
data\resultDesc	String	When result = ok Then this field will indicate the number of clients that are being returned When result = error Then this field will indicate a description of the cause of the error. The possible list of error codes and their causes and expected resolutions will be provided later.
data\clients	–	Node contains each of the client nodes that are a match
data\clients\client	–	Node containing client detail information.
data\clients\client\ClientCompanyName	String	As entered in Administration->Agent Mgt
data\clients\client\address1	String	As entered in Administration->Agent Mgt
data\clients\client\address2	String	As entered in Administration->Agent Mgt
data\clients\client\city	String	As entered in Administration->Agent Mgt
data\clients\client\state	String	As entered in Administration->Agent Mgt
data\clients\client\zipcode	String	As entered in Administration->Agent Mgt
data\clients\client\country	String	As entered in Administration->Agent Mgt
data\clients\client\telephone	String	As entered in Administration->Agent Mgt
data\clients\client\fax	String	As entered in Administration->Agent Mgt
data\clients\client\email	String	As entered in Administration->Agent Mgt
data\clients\client\webpage	String	As entered in Administration->Agent Mgt

Function: loadClientLocations

This function provides a necessary service that allows the developer to find the details of custom locations that have been added to the system by your company.



Custom locations are added to specific service location, for example, the IMS OFFICE is a fixed client location located in Eatontown, NJ. If you search for client locations in another city, the IMS OFFICE will not be returned.

Parameter	Possible Values	Description
clientkey	String	Key Provided by IMS to authenticate a web service request
username	String	Valid username of a user this webservice is impersonating. Permissions (Buy/Sell Rate Visibility) will be applied by username
password	Encrypted String	Password for the above user encrypted with the Private Key provided by IMS
servicelocationname	String	Name of the service location that the client location should be located within.
clientlocationname	String	Name provided by you the client for your location.

The output XML from the loadClient2List function will validate against the below XSD:

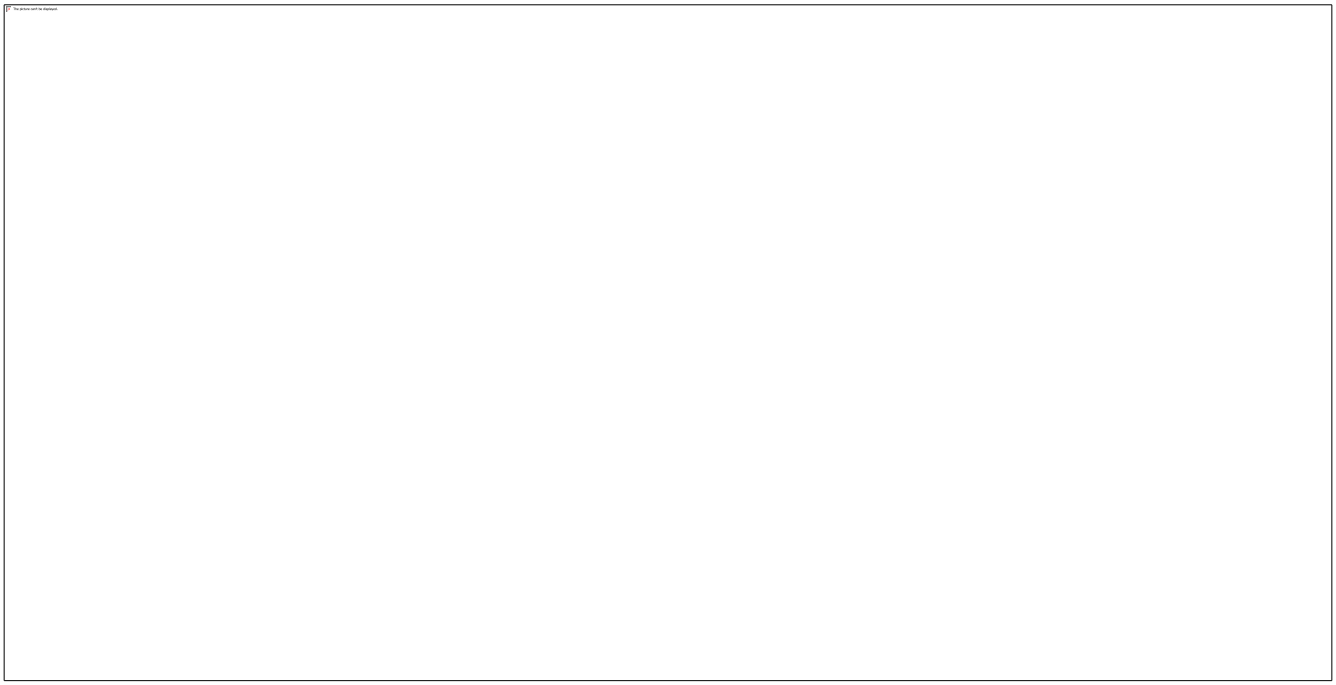
```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="data">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="result" type="xs:string" />
        <xs:element name="resultDesc" type="xs:string" />
        <xs:element name="clientlocations">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="clientlocation">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="servicelocationname" type="xs:string" />
                    <xs:element name="clientlocationname" type="xs:string" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The following table explains the nodes in the XML document:

Element/Node	Possible Values	Description
data	--	Root Node
data\result	ok error	ok – Indicates that the request was successful and clients have been returned error – Indicates that the request was unsuccessful due to some parameters or system error
data\resultDesc	String	When result = ok Then this field will indicate the number of clients that are being returned When result = error Then this field will indicate a description of the cause of the error. The possible list of error codes and their causes and expected resolutions will be provided later.
data\clientlocations	–	Node contains each of the client location nodes that are a match
data\clientlocationss\clientlocation	–	Node containing client location information.
data\clientlocationss\clientlocation\clientlocationname	String	Name of your location
data\clientlocationss\clientlocation\servicelocationname	String	Name of the IMS service location where your location exists.

Function: submitDispatchXML

This function allows your company to submit all of the information necessary to submit a dispatch to IMS and retrieve an IRIS dispatch confirmation number in return. This function provides the functionality as seen on the IMS website under IRIS -> View Quotes --> "Go" (to send a dispatch) and it would be helpful to the developers to be familiar with the dispatch process. The XSD file "DispatchXML.xsd" provided with this documentation details the format of the XML string that must be submitted in order to a dispatch to be processed.



Each dispatch consists of three key peices of information. The "Header" information provides IMS with the details of the voyage/vessel for the container(s) being dispatched. The "Locations" section provides IMS with the detailed origin, destination and empty return information. The "Containers" section provides IMS with the detailed information, including container number, size, and type being dispatched. Each dispatch can container multiple containers provided that all of the containers are being moved between the same "Locations" and are being moved on vessel/voyage provided in the "Header" information.

The submitDispatchXML function takes the following parameters as input:

Parameter	Possible Values	Description
clientkey	String	Key Provided by IMS to authenticate a web service request
username	String	Valid username of a user this webservice is impersonating. Permissions (Buy/Sell Rate Visibility) will be applied by username
password	Encrypted String	Password for the above user encrypted with the Private Key provided by IMS
xmlString	String	XML document as a string. The XML must validate against the DispatchXML.xsd schema.

The input XML schema, as provided in DispatchXML.xsd, must validatie against the following XSD:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="xml">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="dispatch">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="version" maxOccurs="1" minOccurs="1"/>
              <xs:element name="header" maxOccurs="1" minOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="quotenbr"       minOccurs="1" />
                    <xs:element name="ix"             minOccurs="1" />
                    <xs:element name="client2"        minOccurs="0" />
                    <xs:element name="dispatchbranch" minOccurs="1" />
                    <xs:element name="clientref"      minOccurs="1" />
                    <xs:element name="shippingline"   minOccurs="1" />
                    <xs:element name="vessel"         minOccurs="1" />
                    <xs:element name="voyage"         minOccurs="1" />
                    <xs:element name="eta"            minOccurs="0"/>
                    <xs:element name="etd"            minOccurs="0" />
                    <xs:element name="cutoff"         minOccurs="0" />
                    <xs:element name="erd"            minOccurs="0" />
                    <xs:element name="bol"            minOccurs="0" />
                    <xs:element name="booking"        minOccurs="0" />
                    <xs:element name="inbond"         minOccurs="0" />
                    <xs:element name="osport"         minOccurs="0" />
                    <xs:element name="remarks"        minOccurs="1" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="locations" maxOccurs="1" minOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="p1" maxOccurs="1" minOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="type" maxOccurs="1" minOccurs="1" />
                          <xs:element name="name" maxOccurs="1" minOccurs="1" />
                          <xs:element name="clientloc" maxOccurs="1" minOccurs="0">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="name" />
                                <xs:element name="address1" />
                                <xs:element name="address2" />
                                <xs:element name="citystate" />
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                          <xs:element name="clientcontact" maxOccurs="1" minOccurs="0">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="firstname"  minOccurs="1" />
                                <xs:element name="lastname"   minOccurs="0" />
```

```
                <xs:element name="phone"      minOccurs="0" />
                <xs:element name="phoneext"   minOccurs="0" />
                <xs:element name="fax"        minOccurs="0" />
                <xs:element name="cell"       minOccurs="0" />
                <xs:element name="email"      minOccurs="0" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="p2" maxOccurs="1" minOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="type" maxOccurs="1" minOccurs="1" />
          <xs:element name="name" maxOccurs="1" minOccurs="1" />
          <xs:element name="clientloc" maxOccurs="1" minOccurs="0">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="name" />
                <xs:element name="address1" />
                <xs:element name="address2" />
                <xs:element name="citystate" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="clientcontact" maxOccurs="1" minOccurs="0">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="firstname"  minOccurs="1" />
                <xs:element name="lastname"   minOccurs="0" />
                <xs:element name="phone"      minOccurs="0" />
                <xs:element name="phoneext"   minOccurs="0" />
                <xs:element name="fax"        minOccurs="0" />
                <xs:element name="cell"       minOccurs="0" />
                <xs:element name="email"      minOccurs="0" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="p3" maxOccurs="1" minOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="type" maxOccurs="1" minOccurs="1" />
          <xs:element name="name" maxOccurs="1" minOccurs="1" />
          <xs:element name="clientloc" maxOccurs="1" minOccurs="0">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="name" />
                <xs:element name="address1" />
                <xs:element name="address2" />
                <xs:element name="citystate" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="clientcontact" maxOccurs="1" minOccurs="0">
```

```
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="firstname"  minOccurs="1" />
                    <xs:element name="lastname"   minOccurs="0" />
                    <xs:element name="phone"      minOccurs="0" />
                    <xs:element name="phoneext"   minOccurs="0" />
                    <xs:element name="fax"        minOccurs="0" />
                    <xs:element name="cell"       minOccurs="0" />
                    <xs:element name="email"      minOccurs="0" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="containers" maxOccurs="1" minOccurs="1">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="container" minOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="containernbr"       maxOccurs="1" minOccurs="1"  />
              <xs:element name="size"               maxOccurs="1" minOccurs="1"  />
              <xs:element name="type"               maxOccurs="1" minOccurs="1"  />
              <xs:element name="ingauge"            maxOccurs="1" minOccurs="0"  />
              <xs:element name="reeftemp"           maxOccurs="1" minOccurs="0"  />
              <xs:element name="reefunit"           maxOccurs="1" minOccurs="0"  />
              <xs:element name="weight"             maxOccurs="1" minOccurs="1"  />
              <xs:element name="weightunit"         maxOccurs="1" minOccurs="1"  />
              <xs:element name="commodity"          maxOccurs="1" minOccurs="1"  />
              <xs:element name="piececount"         maxOccurs="1" minOccurs="0"  />
              <xs:element name="packagingtype"      maxOccurs="1" minOccurs="0"  />
              <xs:element name="seal"               maxOccurs="1" minOccurs="0"  />
              <xs:element name="pickupnbr"          maxOccurs="1" minOccurs="0"  />
              <xs:element name="ramplfd"            maxOccurs="1" minOccurs="0"  />
              <xs:element name="pierlfd"            maxOccurs="1" minOccurs="0"  />
              <xs:element name="hazyn"              maxOccurs="1" minOccurs="1"  />
              <xs:element name="hazunnumber"        maxOccurs="1" minOccurs="0"  />
              <xs:element name="hazclass"           maxOccurs="1" minOccurs="0"  />
              <xs:element name="hazpackaginggroup"  maxOccurs="1" minOccurs="0"  />
              <xs:element name="quotenbr"           maxOccurs="1" minOccurs="1"  />
              <xs:element name="quotebaserate"      maxOccurs="1" minOccurs="1"  />
              <xs:element name="quotefuel"          maxOccurs="1" minOccurs="1"  />
              <xs:element name="quotehaz"           maxOccurs="1" minOccurs="0"  />
              <xs:element name="quoteow"            maxOccurs="1" minOccurs="0"  />
              <xs:element name="quotereef"          maxOccurs="1" minOccurs="0"  />
              <xs:element name="quotecleantruck"    maxOccurs="1" minOccurs="0"  />
              <xs:element name="quoteallinrate"     maxOccurs="1" minOccurs="1"  />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:sequence>
```

```
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

A sample XML structure is provided in the file "SampleDispatchXML.xml" as below:

```
<xml>
 <dispatch>
  <version>1</version>
  <header>
   <quotenbr>1832409</quotenbr>
   <ix>I</ix>
   <client2></client2>
   <dispatchbranch>IMS DEMO - EATONTOWN, NJ</dispatchbranch>
   <clientref>TESTVIAXML</clientref>
   <shippingline>Allianca</shippingline>
   <vessel>testvessel</vessel>
   <voyage>testvoyage</voyage>
   <eta>01/15/2011</eta>
   <bol>TESTbol</bol>
   <inbond>N</inbond>
   <remarks>this is a test</remarks>
  </header>
  <locations>
   <p1>
    <type>CLIENT</type>
    <name>SOME NEW LOC</name>
    <clientloc>
     <name>SOME NEW LOC</name>
     <address1>123 Main Rd.</address1>
     <address2></address2>
     <citystate>Eatontown, NJ</citystate>
    </clientloc>
    <clientcontact>
     <firstname>test</firstname>
     <lastname/>
     <phone/>
     <phoneext/>
     <fax/>
     <cell/>
     <email/>
    </clientcontact>
   </p1>
   <p2>
    <type>IMS</type>
    <name>TBA</name>
   </p2>
   <p3>
    <type>CLIENT</type>
    <name>TEST 2</name>
    <clientcontact>
     <firstname>test</firstname>
     <lastname/>
     <phone/>
     <phoneext/>
     <fax/>
     <cell/>
     <email/>
    </clientcontact>
   </p3>
  </locations>
  <containers>
```

```
    <container>
      <containernbr>TEST123456-1</containernbr>
      <size>20</size>
      <type>DV</type>
      <ingauge></ingauge>
      <reeftemp></reeftemp>
      <reefunit></reefunit>
      <weight>34000</weight>
      <weightunit>LBS</weightunit>
      <commodity>TESTS</commodity>
      <piececount>1</piececount>
      <packagingtype>xml</packagingtype>
      <seal>12345</seal>
      <pickupnbr>99999</pickupnbr>
      <ramplfd>01/10/2011</ramplfd>
      <pierlfd>01/11/2011</pierlfd>
      <hazyn>N</hazyn>
      <hazunnumber></hazunnumber>
      <hazclass></hazclass>
      <hazpackaginggroup></hazpackaginggroup>
      <quotenbr>1832409</quotenbr>
      <quotebaserate>255.00</quotebaserate>
      <quotefuel>40.80</quotefuel>
      <quotehaz>0.00</quotehaz>
      <quoteow>0.00</quoteow>
      <quotereef>0.00</quotereef>
      <quotecleantruck>0.00</quotecleantruck>
      <quoteallinrate>295.80</quoteallinrate>
    </container>
  </containers>
 </dispatch>
</xml>
```

The following table explains the nodes/childnodes/elements in the XML document:

Element/Node	Possible Values	Description
dispatch	--	Root Node for dispatches. Note that if you are submitting multiple dispatches with one request, you can repeat this node.
dispatch\version	1	The IMS Dispatch Webservice must be able to account for future changes to dispatches. The version element will allow the service to validate the XML string based on multiple requirements that may arrise in future versions. Currently, the webservice only accepts version “1”.
dispatch\header	--	This parent node will contain all of the elements that are part of the header. Each node in the header is outlined below.
dispatch\locations	–	This parent node will contain all of the elements that are part of the location information. Each child node and elements that is part of the “Locations” section is outlined below.
dispatch\containers	–	This parent node will contain one or more container nodes and elements detailing the container to be dispatched..

The minimal structure is thus:

```
<xml>
  <dispatch>
    <version>#</version>
    <header>...</header>
    <locations>...</locations>
    <containers>...</containers>
  </dispatch>
</xml>
```

Within the "Header" of the dispatch:

Element/Node	Possible Values	Description
dispatch\header\quotenbr	Int	Must be a valid, unexpired IMS QuoteNbr for your company for moves matching the points of this dispatch. The QuoteNbr should have been retrieved from the getQuote() function. Each container being dispatched will require a quote number of it's own that matches the origin, destination and empty points from the header.
dispatch\header\ix	I X	I – Import X – Export When sending an Import (I), P1 is the Origin, Port/Ramp/CY, of the move. P2 is the inland destination, Cargo Delivery, and P3 is the Empty Return. When sending an Export (X), P1 is the Empty Pickup, P2 is the Inland Cargo Pickup Location and P3 is the Port/Ramp/CY destination.
dispatch\header\client2	String	Valid ClientCompanyName from a loadclient2List function call. Leave empty is this dispatch is not being sent on behalf of an agent.
dispatch\header\dispatchbranch	String	Valid Branch Name when the dispatch is originating. The list of valid branch names can be determined from logging in to the IMS website.
dispatch\header\clientref	String	Your reference number
dispatch\header\shippingline	String	Valid shippinglinename from the loadShippingLines functions. This is the shipping line moving your container. If the shipping line being used is not listed, please contact IMS.
dispatch\header\vessel	String	Name or Code for the Shipping Line's vessel.
dispatch\header\voyage	String	Name or Code for the Shipping Line's Vessel's Voyage.
dispatch\header\eta	Date (mm/dd/yyyy)	**Only required on Imports**. Estimated Time of Arrival. Valid date today or in the future in the format of mm/dd/yyyy. This is the date that the vessel is expected to arrive in port. Leave empty for Exports.
dispatch\header\etd	Date (mm/dd/yyyy)	**Only required on Exports**. Estimated Time of Departure. Valid date today or in the future in the format of mm/dd/yyyy. This is the date the vessel is expected to leave the port. Leave empty for Imports
dispatch\header\cutoff	Date (mm/dd/yyyy)	**Only required on Exports.** Valid date today or in the future in the format of mm/dd/yyyy. This is the last date that a container can arrive at the port/ramp/cy in order to make it onto this ship/train.

dispatch\header\erd	Date (mm/dd/yyyy)	**Only required on Exports.** Valid date today or in the future in the format of mm/dd/yyyy. This is the earliest date that a container can be dropped back at the port/ramp/cy.
dispatch\header\bol	String	**Only required on Imports.** Bill of Lading.
dispatch\header\booking	String	**Only required on Exports.** Booking Number.
dispatch\header\inbond	Y N	**Only required on Imports.**
dispatch\header\osport	String	**Only required on Exports.** Name of overseas port.
dispatch\header\	String	General notes for this dispatch.

Within the "locations" of the dispatch, you must provde a node for "p1", "p2" and "p3". Each location within the locations element has the same properties.

Element/Node	**Possible Values**	**Description**
dispatch\locations\p#\type	IMS CLIENT	If you are using a pre-defined, fixed IMS Port/Ramp/CY Location, specify IMS. If you will be sending your own location name, address and contact information, specify CLIENT. If the location is not yet known, specify the type as **IMS** and set the below name to **TBA**.
dispatch\locations\p#\name	String	If type is IMS – this must be a valid IMS location name from the locationname field of the locSearch function. If type is CLIENT – this should be the name of an existing Client Location Name from the clientlocationname list provided in loadClientLocations. If you want to add a new client location, leave this field empty. If the location name is not yet known, specify the name as **TBA** and the type as **IMS**.
dispatch\locations\p#\clientloc	--	This parent node will contain all of the elements that are part of the client location being sent.
dispatch\locations\p#\clientloc\name	String	Valid Client Location Name from the clientlocationname list provided in loadClientLocations or any new location name being added to the system.
dispatch\locations\p#\clientloc\address1	String	Client location address line 1
dispatch\locations\p#\clientloc\address2	String	Client location address line 2
dispatch\locations\p#\clientloc\citystate	String	Valid US city and state in the format of: City, State Abv for example: Eatontown, NJ or New York, NY
dispatch\locations\p#\clientcontact	--	This parent node will contain all of the elements that are part of the client location contact being sent. **This node is only required if location type is CLIENT.**.
dispatch\locations\p#\clientcontact\firstname	String	Required
dispatch\locations\p#\clientcontact\lastname	String	Optional
dispatch\locations\p#\clientcontact\phone	String	Phone or Email is required.
dispatch\locations\p#\clientcontact\phoneext	String	Optional
dispatch\locations\p#\clientcontact\fax	String	Optional
dispatch\locations\p#\clientcontact\cell	String	Optional
dispatch\locations\p#\clientcontact\email	String	Phone or Email is required.

Please remember these elements should repeat for p1, p2 and p3.

Within the "containers" of the dispatch, multiple containers can be dispatched in a single web service request provided that all containers and quotes have the same locations and same header information:

Element/Node	Possible Values	Description
dispatch\containers	--	Root Node for containers.
dispatch\containers\container	–	Root node for each container being dispatched. This section can repeat for multiple containers to be sent in one request.
dispatch\containers\container\containernbr	String	Must be a valid container number in the format of **AAAA######-#** For example: ABCD123456-7 = VALID 1A2B3C4E5F6G = INVALID For Exports Only, this can be **TBA**.
dispatch\containers\container\size	20 40 45	Valid container size.
dispatch\containers\container\type	Valid 2 letter type code from list to the right	DV - DRY VAN CONTAINER HC - HIGH CUBE CONTAINER OT - OPEN TOP CONTAINER RF – REFRIGERATED RH - REEFER / HIGH CUBE FL - FLAT RACK *** Note – Flat Rack containers dispatched online must be in-gauge.**
dispatch\containers\container\ingauge	I O	If type is FL, then in-gauge must be I to dispatch online. For out of gauge dispatches, please contact IMS Operations. For all other container types, this element can be left empty.
dispatch\containers\container\reeftemp	Number	Temperature of container if type is RF or RH. Can be left empty for all other container types.
dispatch\containers\container\reefunit	C F	Unit of measure
dispatch\containers\container\weight	Number	Weight of container. For Exports, if weight is unknown, use UNK – Unknown (not overweight) OVW – Unknown (overweight)
dispatch\containers\container\weightunit	LBS KG	Unit of measure
dispatch\containers\container\commodity	String	**Required.** Description of container contents
dispatch\containers\container\piececount	Number	**Optional.** Number of items in container
dispatch\containers\container\packagingtype	String	**Optional.** Description of how cargo is packed into container, for example drums, bags, cartons,pallets, boxes, etc.
dispatch\containers\container\seal	String	**Optional.** Seal number
dispatch\containers\container\pickupnbr	String	**Optional.** Pickup number
dispatch\containers\container\ramplfd	Date (mm/dd/yyyy)	**Optional.** Ramp Last Free Day
dispatch\containers\container\pierlfd	Date (mm/dd/yyyy)	**Optional.** Pier Last Free Day

dispatch\containers\container\hazyn	Y N	**Required.** If hazardous then Y, otherwise N.
dispatch\containers\container\hazunnumber	String	**Required if hazyn is Y.** UN Number
dispatch\containers\container\hazclass	String	**Required if hazyn is Y.** Haz Class
dispatch\containers\container\hazpackaginggroup	String	**Required if hazyn is Y.** Haz Packaging Group
dispatch\containers\container\quotenbr	Number	**Valid IMS Quote Number.** Use the getQuote function to get a valid IMS Quote for the size/type/weight/haz/ref/locations being dispatched. The system will make a best effort to validate the quote against the dispatch, IMS Operations will contact you if there are any discrepancies between the quote number sent and the validity of the quote number used for the move options sent.
dispatch\containers\container\quotebaserate	Number	**quoteAmt from getQuote function.** You must echo back the quote amounts as confirmation that you have seen and are accepting the quote for the specified amount. See rate terms and conditions.
dispatch\containers\container\quotefuel	Number	**fuelFees from getQuote function.** You must echo back the fuel amounts as confirmation that you have seen and are accepting the fees for the specified amount. See rate terms and conditions.
dispatch\containers\container\quotehaz	Number	**hazardousFees from getQuote function.** You must echo back the haz amounts, or 0.00, as confirmation that you have seen and are accepting the fees for the specified amount. See rate terms and conditions.
dispatch\containers\container\quoteow	Number	**OverweightFees from getQuote function.** You must echo back the overweight amounts as confirmation that you have seen and are accepting the fees for the specified amount. See rate terms and conditions.
dispatch\containers\container\quotereef	Number	**reeferFees from getQuote function.** You must echo back the refrigerated fee amounts, or 0.00, as confirmation that you have seen and are accepting the fees for the specified amount. See rate terms and conditions.
dispatch\containers\container\quotecleantruck	Number	**cleanTruckFees from getQuote function.** You must echo back the clean truck fee amount as confirmation that you have seen and are accepting the fees for the specified amount. See rate terms and conditions.
dispatch\containers\container\quoteallinrate	Number	**Sum of base + fuel + haz + overweight + reefer + clean truck.** This sum must match the all in rate from the getQuote function as well as the sum of the dispatch being sent.

Available Sample Clients

IMS provides sample clients in two well established programming languages; PHP and VB.NET. Developers who are familiar with languages such as a PERL, Python, C, C++ should be able to recognize the PHP code structure and develop an application based upon that sample. Developers who are familiar with languages such as a C# or Java should be able to develop an application based upon the VB.NET sample. IMS may or may not be able to provide additional development support for languages other than PHP or VB.NET. As of this time, these languages are the only two languages officially supported by IMS and tested with the IMS Web Service although any language that can access SOAP web services should work correctly and without issue.

PHP Sample

The PHP Sample uses the ZEND Framework Zend_Soap_Client class in order to establish a SOAP Client connection. The sample includes six sample function calls to demonstrate several possible usages of the IMS Web Service as well as the function needed to encrypt a user's password using the provided private key.

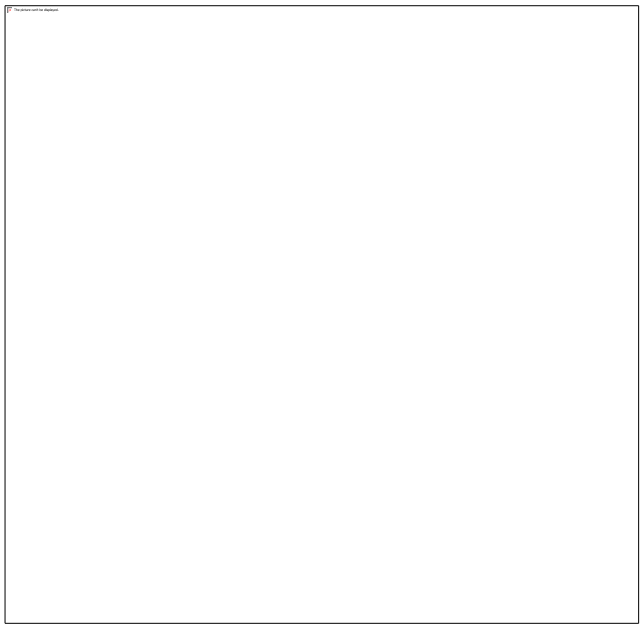
VB.NET Sample

The VB.NET Sample uses a web reference to the IMSService to establish a SOAP Client connection. The sample includes sample functions to demonstrate each of the available functions as well as the function needed to encrypt a user's password using the provided private key.

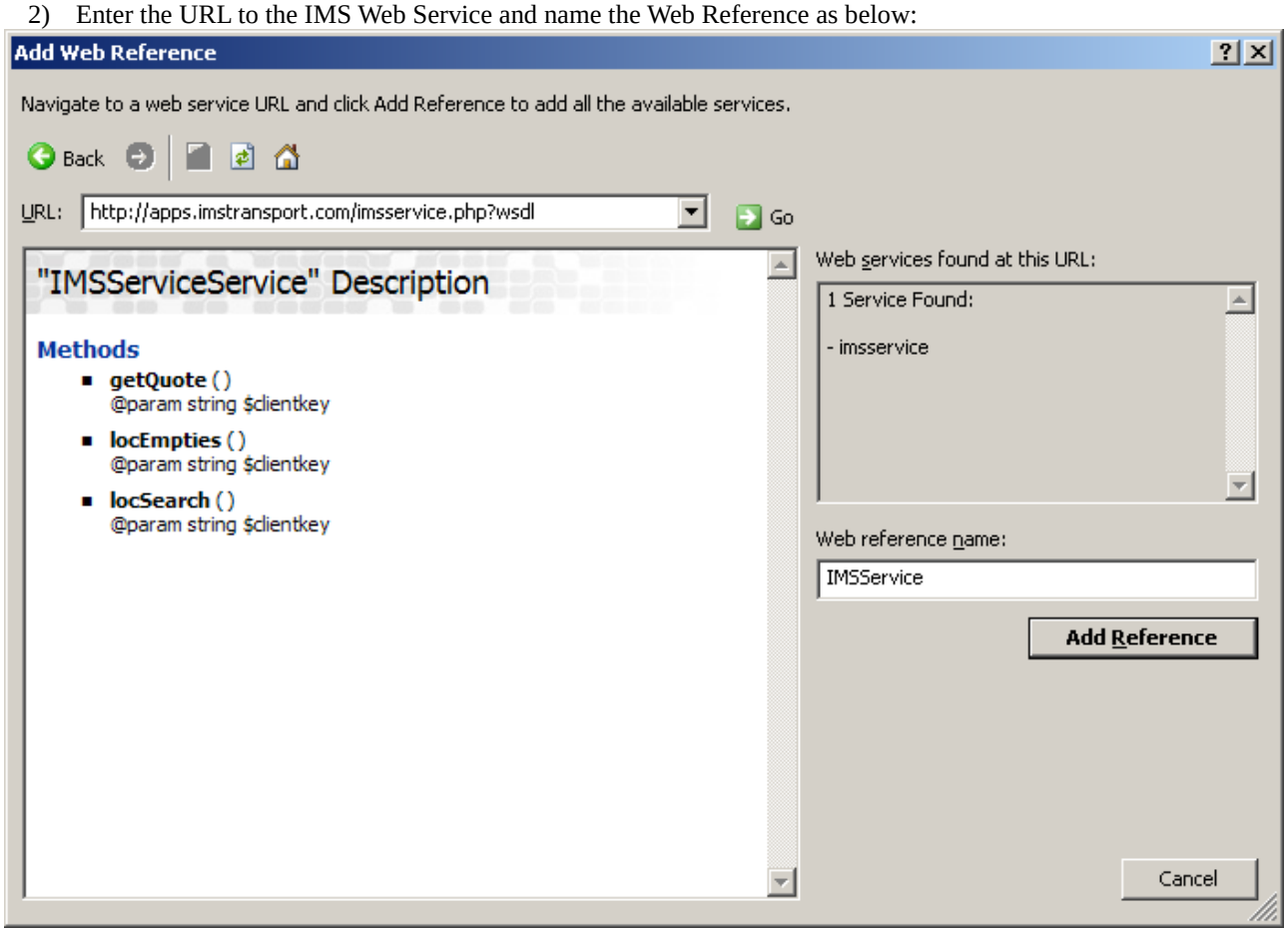
Adding a Web Reference to a VB.NET Client

This section assumes you have a basic understanding of VB.NET and explains how to create a connection to the IMS Web Service. Again, the section assumes that you are reasonably familiar with VB.NET programming and is not a complete sample or a programming guide.

1) Create a new project and add a Web Reference to your project:



2) Enter the URL to the IMS Web Service and name the Web Reference as below:



When adding the Web Refence, enter the **URL** and click the **Go** button. Then change the **Web reference name** to IMSService. You should see the supported functions documented in this guide as available methods as the screen shot demonstrates. Click Add Reference when you have confirmed that you are connected to the IMS Web Service.

Java Sample

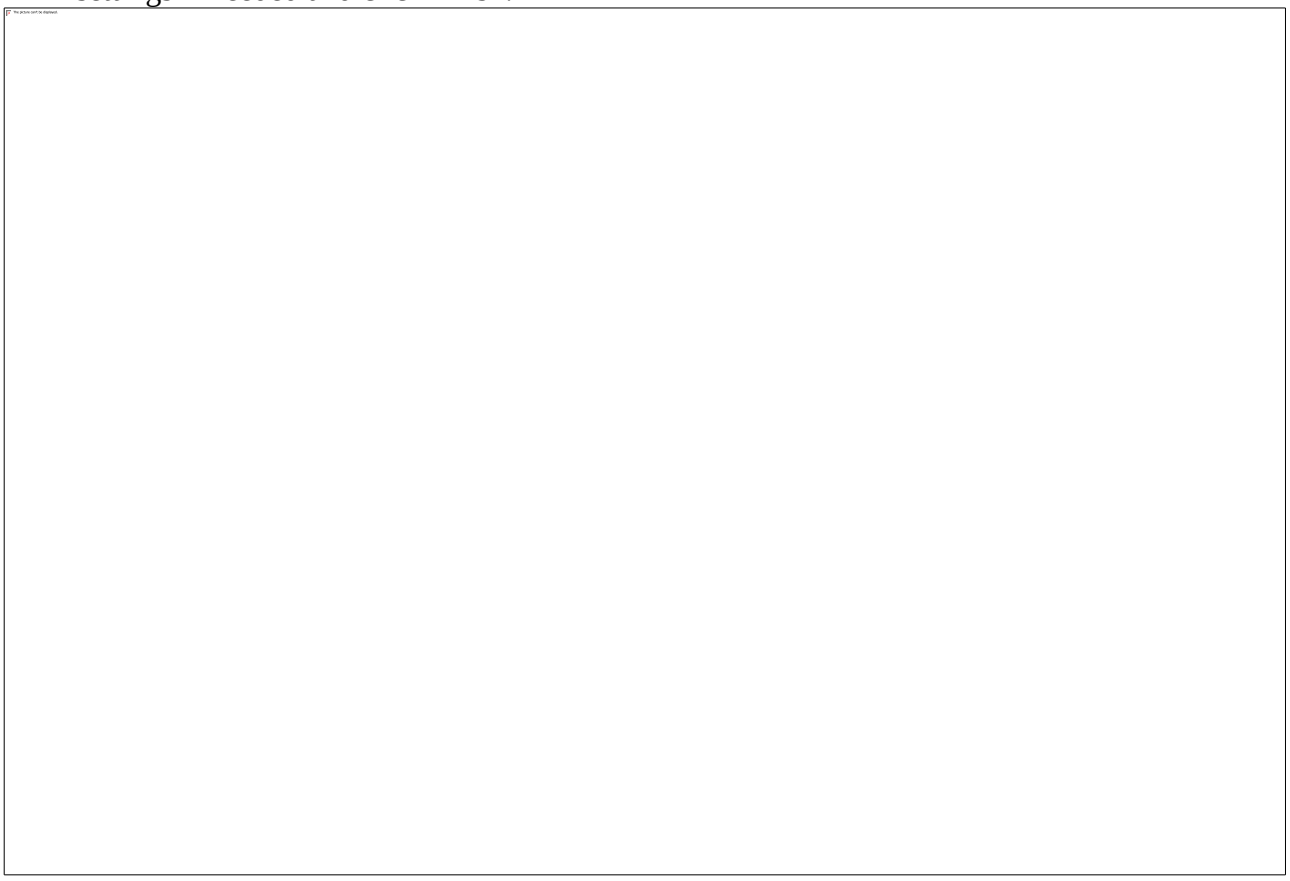
This is a simple example of how to utilize the functions to get a quote and then dispatch one container. We provided an executable jar file which will run the sample program and the source code. The sample was developed using Netbeans 7.0.1 (http://netbeans.org/downloads/index.html).

Adding the web service to your Java program

1) Start up Netbeans and create a new project (standard java app option is fine). We named ours webServiceApp.

2) Once project is created, right click on name, select 'new', choose 'Web Service Client...'

3) Choose 'WSDL URL' option, add web service address (do **NOT** include the http), change proxy settings if needed and click finish.



4) You will now see 'Web Service References'. Expand the options and you will see the functions provided from the web service.

5) Click and drag whatever functions you need to use into the program and Netbeans will generate the code needed. You can now call the function as you would any other function.

IMS FasTrak Move Status Notification Service

The IMS FasTrak Move Status Notification Service provides a method to obtain FasTrak Track & Trace information via a systematic method, on a company level, for all containers dispatched to IMS. The service is a “push” type service in which IMS will generate an standardized XML document, or translated text/HTML document, and “push” the notification to you via FTP, E-mail or FTP and E-mail. IMS can push the notification files to your own FTP server if you provide the login credentials to us, or we can push the files to our own FTP server where you can retrieve them. Each XML Notification file may contain one or more records for one or more containers. The service is intended for clients who will then process the status information into a system to present to users. It is not intended as a method to notify individuals users of container movement.

The IMS FasTrak Move Status Notification Service must be activated independently of the IMS Web Service. Please contact IMS if you wish to have this service enabled for your company account.

XML Schema for Move Status Notification Service

Below is the XSD for a status event notification.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="xml">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="record">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="id" type="xs:string" />
              <xs:element name="clientcompanyid" type="xs:unsignedInt" />
              <xs:element name="imsnbr" type="xs:unsignedInt" />
              <xs:element name="containernbr" type="xs:string" />
              <xs:element name="booking" type="xs:string" />
              <xs:element name="bolfull" type="xs:string" />
              <xs:element name="clientrefnbr" type="xs:string" />
              <xs:element name="irisreferenceid" type="xs:string" />
              <xs:element name="movetype" type="xs:string" />
              <xs:element name="statuscode" type="xs:string" />
              <xs:element name="eventdate" type="xs:string" />
              <xs:element name="statusdate" type="xs:string" />
              <xs:element name="statusnote" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The following table details the return elements:

Element/Node	Possible Values	Description
xml	--	Root Node – occurrs only once
xml\record	--	Starts a new status notification record
xml\record\id	Int	Unique Record Id
xml\record\clientcompanyid	Int	Your IMS Company ID. All notifications sent to your company should container your ID.
xml\record\imsnbr	Int	IMS Assigned Number for this move
xml\record\containernbr	String	Container number or “TBA”
xml\record\booking	String	Booking Number – only sent on Export containers
xml\record\bolfull	String	Bill Of Lading – only sent on Import containers
xml\record\clientrefnbr	String	Client Reference number
xml\record\irisreferenceid	Int	IMS IRIS Confirmation number
xml\record\movetype	String	I (Import) or X (Export)
xml\record\statuscode	String	Status from IMS Status Code List (next section)
xml\record\eventdate	String	Date the event was logged by IMS
xml\record\statusdate	String	Date the event occurred
xml\record\statusnote	String	Remarks associated with the note

IMS Status Code List

Below is the list of possible status codes that will be sent in the Status Notification Process. Additional status codes that are currently available on the website FasTrak reports may be available upon request.

Status	Apt Status	Clearance Status	Gate Status
APPT RESCHEDULED	Y	N	N
APPT SET	Y	N	N
CANCELLED	N	N	N
CARGO DELIVERED	Y	N	N
CARGO PICKED UP	Y	N	N
CONTAINER ARRIVED	Y	N	N
CONTAINER DELIVERED	Y	N	N
CONTAINER LOADED	Y	N	N
EMPTY IN GATE	N	N	Y
FULL IN GATE	N	N	Y
FULL OUT GATE	N	N	Y
MOVE CANCELLED	Y	Y	Y
MOVE CREATED	N	N	Y
RAMP FULL IN GATE	N	N	Y
RAMP FULL OUT GATE	N	N	Y
TRUCK FULL OUT GATE	N	N	Y
YARD FULL IN GATE	N	N	Y
YARD FULL OUT GATE	N	N	Y

Available IMS Move Status Notification Delivery Options

There are three move status notification options:

FTP:	XML documents can be sent to your FTP. You must provide IMS with the FTP Server name, target directory, username and password.
E-Mail:	The XML document is sent as an attachment to this email address. The body of the email will containe an HTML translated version of the XML document. (see below for sample XSLT to translate XML to HTML)
FTP and E-Mail:	XML documents can be sent to your FTP. You must provide IMS with the FTP Server name, target directory, username and password.

XSLT for Move Status Notification XML

The below XSLT will translate the IMS Move Status Notification XML document into an HTML presentable table.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
  <html>
   <body>
    <h2>IMS Move Status Updates</h2>
    <table border="1">
     <tr bgcolor="#E0E0E0">
      <th>IMSNBR</th>
      <th>CONTAINER NUMBER</th>
      <th>BOL</th>
      <th>BOOKING</th>
      <th>CLIENTREF</th>
      <th>IRISREF</th>
      <th>MOVETYPE</th>
      <th>STATUS CODE</th>
      <th>STATUS DATE</th>
      <th>EVENT DATE</th>
      <th>STATUS NOTE</th>
     </tr>
     <xsl:for-each select="xml/record">
      <tr>
       <xsl:choose>
        <xsl:when test="position() mod 2 = 1"><xsl:attribute name='bgcolor'>#FFFFFF</xsl:attribute></xsl:when>
        <xsl:otherwise><xsl:attribute name='bgcolor'>#E0E0E0</xsl:attribute></xsl:otherwise>
       </xsl:choose>
       <td><xsl:value-of select="imsnbr" /></td>
       <td><xsl:value-of select="containernbr" /></td>
       <td><xsl:value-of select="bolfull" /></td>
       <td><xsl:value-of select="booking" /></td>
       <td><xsl:value-of select="clientrefnbr" /></td>
       <td><xsl:value-of select="irisrefnbr" /></td>
       <td><xsl:value-of select="movetype" /></td>
       <td><xsl:value-of select="statuscode" /></td>
       <td><xsl:value-of select="statusdate" /></td>
       <td><xsl:value-of select="eventdate" /></td>
       <td><xsl:value-of select="statusnote" /></td>
      </tr>
     </xsl:for-each>
    </table>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

XML File Naming Convention

The standard naming convention for XML Notification files will be:

STATUS_**[PROVIDER]_[CLIENT]_[FILEDATETIME]**.xml

For example, you will receive a file such as:

STATUS_26912_12345_20111215161245.xml

Which will indicate:

"STATUS" notification file

Sent By: IMS (26912)

Sent To: Client (12345)

File Date: 12/15/2011 at 4:12:45PM ET

E-Mail Notification Requirements

If you opt to receive status notifications via e-mail, the emails will be sent with the standard message subject of:

IMS Shipment Notification

You can provide one or more email addresses for notifications to be sent. The email will be sent from:

operations@imstransport.com

Options and Customization

IMS offers several levels of customization for this process.

- You can select which status codes to receive so that not all status codes are sent.
- IMS can change file name formats
- IMS can apply a custom XSLT to the XML document before it is sent in order to provide a customized Text file or an XML file in another format. For example, IMS Status Notifications can be sent in the format below:

```
<?xml version="1.0" ?>
<VendorTracking xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <VendorTrackingEnvelope>
  <SenderID>IMS</SenderID>
  <ReceiverID>CLIENT</ReceiverID>
  <Password>PASSWORD</Password>
  <Type>VendorTracking</Type>
  <Version>1</Version>
  <EnvelopeID>1234567890</EnvelopeID>
 </VendorTrackingEnvelope>
 <VendorTrackingDetails>
  <BookingNumber>YOURBOOKINGNBR</BookingNumber>
  <BillofLadingNumber />
  <CustomersIdentifier>IMSNUMBER</CustomersIdentifier>
  <ContainerNumber>CONTAINERNUMBER</ContainerNumber>
  <VendorReference>CLIENTFILEREF</VendorReference>
  <Mode>O</Mode>
  <TypeOfMove>F</TypeOfMove>
  <StatusDetails>
   <CurrentStatus>
    <StatusCode>207</StatusCode>
    <StatusName>FULL IN GATE</StatusName>
    <StatusDate>201112140832</StatusDate>
    <StatusTime>09:10:00</StatusTime>
   </CurrentStatus>
   <NextStatus>
    <StatusCode />
    <StatusName />
   </NextStatus>
  </StatusDetails>
  <StatusLocationCode />
  <StatusLocationName>IMSLOCATIONNAME</StatusLocationName>
  <CommentDetails>
   <Remarks>STATUS NOTES</Remarks>
  </CommentDetails>
 </VendorTrackingDetails>
</VendorTracking>
```