TextPower

Alert and authenticate using text messaging

TECHNICAL DOCUMENTATION
FOR:

Advanced SOAP Interface Specification

TextPower, Inc.
www.TextPower.com
Twitter.com/TextPower
Support@TextPower.com
888-818-1808

SOAP Advanced Interface

This manual describes Version 3 of the Advanced SOAP interface to TextPower, Inc. (TPI) services. There are multiple levels of SOAP interfaces. Only the Advanced Interface is described in this manual. Access to the SOAP interfaces does not come automatically with TPI service. Access to each SOAP level must be specifically requested. Contact your account representative.

The Advanced SOAP interface is intended to allow the application developer to build very sophisticated applications based upon the TPI switch and services. Note that this Version 3 now supplies all dates in the time zone of the user rather than UTC as previous versions did.

Table of Contents

Overview

This manual will not attempt to describe the general SOAP interface and structure. It is expected that the reader already knows this. This manual provides only supplementary information that is not found in the WSDL descriptions. This interface along with the other SOAP interfaces allows the TextPower (TPI) customer to build extremely sophisticated messaging applications. Building a SOAP interface should not be undertaken unless the developer has the proper development tools to auto-generate the necessary interface code and the expertise to develop the SOAP application.

TPI Advanced SOAP functions return either an XML fragment (XML node), structure or a DataSet. A SOAP header is required on all calls and the header information is identical on all calls. Structural errors involving invalid credentials, incorrect parameter values, etc. return an XML fragment with Errors as the base node if the output type is XMLNode. If the return is a structure, there will be an errorDescr element in the structure. If the output type is DataSet, structural errors return a SOAP exception. SOAP exceptions are supported. SOAP exceptions represent internal program faults for functions that return a XMLNode but not necessarily for functions that return a DataSet. The return of an Errors node or a SOAP Exception usually represents a detectable error in the information supplied. Please check the information that you are supplying to correct this type of problem. For exceptions that do not represent structural or credentials problems, contact TPI support.

For methods that either take a DataSet as input or return one, it is important to reduce the number of rows transmitted as much as possible to keep overhead, bandwidth, etc., minimized. To minimize row transmission loads, follow these guidelines.

1. For GetMOData and GetMTData, utilize the LastReceiveID or LastSendID parameters, setting them to the maximum value that you received on your last request. Setting them at zero all the time will just keep retrieving the same rows again and again.
2. For Send List functions, send the number list up one time and leave it, only adding or deleting deltas to the list. Sending an entire list up each time you want to do a send when the list is essentially the same as the last time you sent a list needlessly uses processing power and slows down your own server.

In this manual, the term MT means Message Terminating. That means a message that you sent to a phone. MO means Message Originating. That means a message a phone sends to you. The terms terminating and originating always are from the perspective of the phone.

WSDL and Header

The WSDL document for the Advanced SOAP interface is located at
http://www.textpower.com/TPIServices/AdvancedMessageServicesV3.asmx?wsdl

or
https://secure.textpower.com/TPIServices/AdvancedMessageServicesV3.asmx?wsdl

The address for the SOAP calls on the advanced interface is, of course:
http://www.textpower.com/TPIServices/AdvancedMessageServicesV3.asmx

or

https://secure.textpower.com/TPIServices/AdvancedMessageServicesV3.asmx

The SOAP header on all calls has four parameters as described in the WSDL. All header parameters are of type String.

Parameter	**Meaning**
UID	Required. The value of your UserID as assigned by TPI.
PWD	Required. The value of your Password as assigned by TPI or set by you from the TPI web site.
Campaign	The name of the campaign you are using for this call. If you have only one campaign, this parameter may be a zero length string.
Keyword	The keyword in your campaign that you are using for this call. If you have only one keyword for your campaign or do not have a keyword in your campaign, this parameter may be a zero length string.

The majority of users has only one campaign each and only one keyword and will be able to leave the Campaign and Keyword parameters empty.

Function Notes

Type specifications for each call and the parameter data types must be taken from the WSDL. The basic purpose of each function is also described in the WSDL. This section provides additional information on the meaning of each call, its parameters and return values along with an example of the returned XML node or the schema of the returned DataSet. All DataSets for the Advanced SOAP interface have one or two tables within them. TPI Advanced SOAP is coded using the Microsoft .NET technology. If you are

using PHP or some other technology for your server, use libraries that are designed to work with .NET DataSets.
Note that in the WSDL, the Integer type is referred to as int, the Date type is referred to as dateTime and the XmlNode and DataSet types are complex types. They are the same thing and may be used interchangeably. Sample output XML has been formatted for display on this page using line feeds and indents. These do not appear in the actual XML sent from the TPI switch. The Level parameter for output nodes denotes the tag nesting level. The root or base node is 1.

CellNumber Parameter Note: During development of projects, it is sometimes desirable to send messages just for the purpose of testing an interface but which you don't want delivered to a real phone. The TPI SOAP interfaces allow for that case. To send a message without actually having it delivered anywhere, use numbers in the reserved fictitious movie range. These numbers are not cell phones nor are they even real numbers. The TPI SOAP interfaces will accept them and not give you an error. Your logs will be identical to those of real numbers and show these numbers but **no actual message will be delivered**! If your account requires a carrier to be supplied, you must still supply a valid carrier code with fictitious sends but it will be ignored. Fictitious sends still count against any account limits you may have on your account. Numbers in the reserved fictitious movie range are of the form:
NXX55501XX where N is a number from 2-9 and X is a number from 0-9.

Date and Time formats: Note that the dates and times in the Advanced Specification follow the following rules:

1. Simple values of both inputs and outputs are expressed in Universal Time.
2. Date values in Datasets, both input and output, are expressed in Central Time.

Send Lists

This document will refer to Send Lists in several places. These items are central to the TPI system. One of the keys to using the Advanced SOAP interface is to understand what these items are and how to use them. Using these lists is the fastest, highest capacity way to send to large blocks of numbers. If you have a large block of numbers to send to and you want to send the same message to all of them, using lists will get you the highest capacity that TPI offers and will generally far exceed the capacity available by using the SendSMS or SendSMSFull functions on the basic SOAP interface.

Subscription Lists

Subscription lists were used for premium message services. As US carriers have almost totally shut down premium message services and TextPower no longer offers them, all Subscription List API calls have been deleted in Version 3.

Send Lists

Send Lists are lists of numbers that you want to send to. You may have as many of these as you want and can slice and dice your lists for various purposes.
If you are an Opt-In service, you always have a special Send List called the Client List. The Client List consists of all validly Opted-In numbers to your service. You can send directly to the Client List using the SendToClients function in the basic SOAP interface. You do not have to upload this list. It is always there. When you do a SendToClients, a special Send List will be created. The name will be of the form, "Client-" plus a number. You can find it with GetSendLists command.

If you want to send to just a portion of your Client List or to any large group of numbers if you are a No Opt-In service, you should consider using and uploading a Send List with just those numbers in it. Because of the way traffic is handled internal to the TPI switch, this is much faster than sending numbers up using either multiple mode of SendSMS on the basic SOAP interface or Sender.aspx on the classic interface.

Send Lists are referenced by a List Name. When you use a send list and you are an Opt-In client, all numbers in the send list must be Opted-In under your keyword. If you have only one keyword on your account, the keyword will always be the correct one. If you have multiple keywords on your account, please note the following.

- If you create a send list using credentials for keyword A on your account and then later try to send to that list using credentials for keyword B on your account, all of your sends will fail with a No Opt-In error.
- Whenever you add/merge numbers to a send list, the keyword associated with that list will be updated to the keyword associated with the credentials on that call.

Send Lists have three items that an application can control.

1. Cell Number: The number of the cell phone. Always required.
2. Carrier: The carrier code for the number. If you definitely know the carrier code, upload it with your dataset from the add/merge numbers method. If you don't know the carrier, which is the most common case, leave the carrier code blank(not Null) or put a "0" in it. After the add/merge command is complete, the TextPower system will initiate a background process to discover the carrier for all numbers in the send list that do not have valid carriers. This consists of two phases:
    - All numbers that do not have valid carriers but are in the TextPower carrier cache will have their carriers inserted from the TextPower cache. No carrier lookup charges will occur for this.
    - All numbers that do not have valid carriers and are not in the carrier cache will have their carriers looked up. TextPower's standard lookup charge applies to these lookups. The carriers found are inserted in the TextPower carrier cache so future carrier substitutions come from the cache. This process runs in the

background after the completion of the MergeSendList command. It is rate controlled to a maximum of 100 lookups/minute. The salient point here is that the Send List may not be fully ready for sending until several minutes later. Consider this example for a list of 2000 numbers added via an MergeSendList command.

- Of the 2000 numbers in the list, 1000 have carriers available in the TextPower cache. Those carriers will be inserted within one minute without charge.
- For the remaining 1000 numbers, charged lookups occur. Ten minutes later at a minimum, all carriers will be looked up and the Send List will be fully ready for sending.

3. Comment: An optional 50 character field for application use, for say a subscriber name or any other use the application may have. This field is not used in any sending capacity.

Why is sending to lists faster than using the multiple number mode of SendSMS?

1. You don't have to upload the numbers every time you do a send. You can upload the numbers once into a list and then just reference that list for later sends. This saves you money on row transfer charges and improves the net speed.
2. When the whole package of numbers is known in advance, the TPI system can package them up into an optimal multi-threaded send package that can greatly increase the rate at which the sends take place.
3. The High Capacity, High Reliability and Extended Message Length options are only available when using Send Lists or Subscription Lists.

The Advanced SOAP interface provides methods for uploading send lists and doing adds and deletes to existing lists.

Extended Length Messages

The SendToClients function on the Basic interface and the SendToSendList function on the Advanced interface support extended length messages. You cannot use extended length messages on any other send call! Extended length messages can be up to 800 characters long. Since Cell Phones can take only 160 characters in a single message, an extended length message will be sent in more than one packet. The rules for using Extended Length Messages are:

1. An extended length message must be more than 160 characters in length. For messages <= 160 characters, no check for delimiters will be done and the message will be sent exactly as submitted.
2. The user must provide a delimiter between segments of the message. The delimiter is ;:; (semi-colon, colon, semi-colon). The TextPower system will split the message at the given delimiters and transmit it in bursts of 160 characters or less. The delimiters will not be transmitted.

3. Each segment of the message must be <=160 characters in length.
4. When retrieving the status of sends via the GetSendListMembers function on the Advanced interface, only the SendID of the last message segment sent will appear.

Function Calls

GetMOData

Function GetMOData(Count As Integer, LastReceiveID As Integer) As DataSet

Retrieve MO data for this account. Only user MOs are retrieved. Opt-In and Help MOs are not retrieved.

Input Parameters		
Name	**Data Type**	**Meaning**
Count	Integer	The maximum number of MO records to be retrieved. Zero indicates all.
LastReceiveID	Integer	Retrieve MOs whose ReceiveID is greater than LastReceiveID. If you have made repeated calls to GetMOData, you should set the LastReceiveID to avoid duplicate retrievals and higher row retrieval charges. By using the Count and LastReceiveID parameters, you can limit your individual retrievals to avoid any timeout issues with calls that may return too much data.
Columns Returned		
Column	**DataType**	**Meaning**
ReceiveID	Integer	A unique integer identifying this row. You may use it as a primary key.
CellNumber	String	The cell number of the phone making the MO call.
Keyword	String	The keyword the user used. This will always be one of your assigned keywords.
Message	String	The full text received from the user.
DateReceived	Date	The date and time of the MO in your local time.

GetMTData

Function GetMTData(Count As Integer, LastSendID As Integer, ReturnMessage As Boolean) As DataSet

Retrieve MT data for this account.

Input Parameters		
Name	**Data Type**	**Meaning**
Count	Integer	The maximum number of MT records to be retrieved. Zero indicates all.
LastSendID	Integer	Retrieve MTs whose SendID is greater than LastSendID. If you have made repeated calls to GetMTData, you should set the LastSendID to avoid duplicate retrievals and higher row retrieval charges. By using the Count and LastSendID parameters, you can limit your individual retrievals to avoid any timeout issues with calls that may return too much data.
ReturnMessage	Boolean	If True, return the message sent. If False, do not return the message. Eliminating the message return speeds up the processing. (The sender, by definition, will know the message and thus there is no benefit to returning the message anyway.)
Columns Returned		
Column	**DataType**	**Meaning**
SendID	Integer	A unique integer identifying this row. You may use it as a primary key.
CellNumber	String	The cell number of the phone to which the MT was sent.
Carrier	String	The CarrierCode that was used for the send.
DateSent	Date	The date and time of the MT in your local time.
Status	String	The status received on the send.
SendType	String	A code specifying how the message was sent:

Code	**Meaning**
MESSAGE or MSGX	Sent via a non-queued send off the classic interface
SOAPMESSAGE	Sent via a non-queued

			send off the Basic SOAP interface
		MOQUEUE	Sent via the single message queue
		DSENDER	Sent as a deferred send
		QSENDER	Sent via the mass send queue.
		POST	Posted out to an application and not sent as a message. **Note:** For the POST type, there is no STATUS.
Message	String	The message sent to the user. This column is returned only if ReturnMessage was True.	

GetClientListMembers

```
Function GetClientListMembers() As DataSet
```

A Client List is the list of all cell numbers and/or email addresses Opted-In against a particular short code and keyword. The relevant short code and keyword can be determined from the credentials in the header and does not need to be re-entered in other parameters.

The GetClientListMembers returns a dataset of all clients Opted-In under the account and keyword found from the data in the credentials header. Since the header provides all necessary information, there are no parameters.

The returned dataset will contain one or two tables within it. Table 0 is the list of cell numbers Opted-In and Table 1 is the list of email addresses Opted-In.

Input Parameters		**None**
Columns Returned		
Column	**DataType**	**Meaning**
CellNumber	String	The cell number of the member
Carrier	String	The CarrierCode which is used in the send command. For email addresses in cases where messages must be sent through an SMTP gateway this would not apply.
ActualOpt-InDate	String	The date and time of the Opt-In in your local time.

It is important to keep in mind the difference between a Send List and a Client List. A Client List is simply the list of numbers that could be sent to on one of the bulk send calls. You cannot get the status of sends from a client list since there are no sends directly executed from a Client List. When you do a send from a Client List, as Send List is created from the Client List, which may consist of the whole Client List or only a subset of it. When you do a send, the Send List may be queried to get the status of that particular send. See the other functions in this document for the particulars on how to do that.

This function has no paging ability. TextPower now offers paging ability on retrieval of Client Lists. See below for the new calls on Version 3 that implement this.

GetClientListPageCounts

```
Function GetClientListPageCounts() As ClientCounts
```

GetClientListPageCounts retrieves the total count and the number of pages for taglista and email and cell number optins. For paging purposes, a page consists of 100 cell numbers or email addresses. GetClientListPageCounts needs no parameters because the identity of the list is already contained in the header information. The structure identity can be obtained through the web reference and can be defined in your code that way.

Input Parameters		**None**
Data Returned in Structure ClientCounts		
Element	**DataType**	**Meaning**
TotalCellCount	Integer	The total Cell Numbers in the Client List
PageCellCount	Integer	The total number of 100 record pages in the Client list for Cell Numbers.
TotalEmailCount	Integer	The total Email Addresses in the Client List
PageEmailCount	Integer	The total number of 100 record pages in the Client list for Email Addresses.
TotalTagCount	Integer	The total number of send tags for this Client List
PageTagCount	Integer	The total number of 100 record pages in the Client list for Send Tags.
errorDescr	String	Blank if no error.

GetNextClientListMembers

Function GetNextClientListMembers(ByVal Page As Integer, RetrieveOption As ListTypes) As DataSet

GetNextClientListMembers retrieves a specific page of numbers or email addresses from your Client List as specified in the header. A page is a set of up to 100 numbers or email addresses.

Input Parameters		
Page		The page number that you wish to have returned. The first page is page 1.
RetreiveOption		Set Retrieve Option to 0 for Cell Numbers and 1 for Email Addresses. The ListTypes enumeration is defined on the web service and should be used.
Columns Returned		
Column	**DataType**	**Meaning**
CellNumber	String	The cell number of the member
Carrier	String	The CarrierCode which is used in the send command. For email addresses in cases where messages must be sent through an SMTP gateway this would not apply.
ActualOpt-InDate	String	The date and time of the Opt-In in your local time.

GetNextSendTags

```
Function GetNextSendTags(ByVal Page As Integer) As DataSet
```

GetNextSendTags retrieves a specific page of send tags for your Client List as specified in the header. A page is a set of up to 100 send Tags.

Input Parameters		
Page		The page number that you wish to have returned. The first page is page 1.
Columns Returned		
Column	**DataType**	**Meaning**
CellNumber	String	The cell number of the member
ShortCode	String	The ShortCode of the number for this Client List
Keyword	String	The Keyword of the number for this Client List
Tag	String	The tag name for this number, short code and keyword
AddDate	String	The date and time of the Add Date of the tag in your local time.

GetSendLists

Function GetSendLists() As XmlNode

Retrieve all the send lists for this account.

Input Parameters(none)		
Output Base Node:		SendLists
Sub Nodes		
Tag	**Level-Type**	**Meaning**
SendList	2-Container	The name attribute of the SendList tag contains the name of the send list. There will be one SendList container for each send list in the account.
Count	3-Integer	The count of the members of this send list.
SendStartTime	3-Date	The starting time of the last send to this list. If the starting time is empty, it indicates that no send has been done to this list.
SendEndTime	3-Date	The ending time of the last send sent to this list. If the SendEndTime is empty it indicates that no finish time has yet been recorded. (If the SendStartTime is empty, there will be no SendEndTime because nothing has been queued to send.) If the SendStartTime contains information but there is no information in the SendEndTime it indicates that the send is still in progress.
Output Example		

```
<SendLists xmlns="">
<SendList Name="Temp-5">
        <Count>6142</Count>
        <SendStartTime>8/7/2007 10:03:02 PM</SendStartTime>
        <SendEndTime>8/7/2007 10:10:27 PM</SendEndTime>
</SendList>
<SendList Name="Temp-7">
        <Count>3</Count>
        <SendStartTime>3/12/2008 10:31:46 PM</SendStartTime>
        <SendEndTime>3/12/2008 10:31:50 PM</SendEndTime>
</SendList>
<SendList Name="Temp-9">
        <Count>1</Count>
        <SendStartTime></SendStartTime>
        <SendEndTime></SendEndTime>
</SendList>
</SendLists>
```

GetSendListMembers

Function GetSendListMembers(ListName As String) As DataSet

Retrieve all the members of the given Send List along with their status.

Input Parameters		
Name	**Data Type**	**Meaning**
ListName	String	The name of the list from which the numbers should be retrieved.
Columns Returned		
Column	**DataType**	**Meaning**
CellNumber	String	The cell number of the member
Carrier	String	The Carrier Code that was used for the send.
Comment	String	An optional comment about this number. The comment field is a free form 50 character field that can be used for multiple purposes such as storing a name.
LastSendStatus	String	The status received on the last send. This field may be NULL if no send has been executed.
LastSendDate	Date	The date and time of the last send. This field may be NULL if no send has been executed.
CarierName	String	The common name of the carrier used
SendID	Integer	A unique identifier for the last send to this number. This identifier can be used later to retrieve delivery status or other functions, if desired. If the last send to this number failed or no sends have been done, the SendID will be 0. If the SendID is 0 then the value of the FinalStatus and Reason fields in meaningless.
FinalStatus	String	The delivery status of the last message in words. This status may not arrive for several minutes after the send. If the phone is off, it could come many hours later.
Reason	Integer	The integer reason code indicating the delivery status. This status may not arrive for several minutes after the send. If the phone is off, it could come many hours later. Successful delivery is either 4 or 6 depending upon the carrier. The number of non-delivery failure codes is too large to mention here.

SendToSendList

```
Function SendToSendList(ListName As String, Msg As String, DelaySend As
Boolean, SendTime As Date) As XmlNode
Send a message to all members of the given list.
```

Input Parameters		
Name	**Data Type**	**Meaning**
ListName	String	The name of the list to which the message should be sent.
Msg	String	The message that you want to send. The Msg parameter for this call supports extended length messages and PowerText functions. The characters ~[]^|{}` are invalid in an SMS message and if present will cause your message to fail. In addition, all characters with code values above 127 are invalid. Note that some applications like Microsoft Word convert things like quotes into what they call smart quotes. These smart quotes are invalid. Regular quotes are fine but not smart quotes. To be safe, never paste message text from a word processor into an SMS message box.
DelaySend	Boolean	If False, send the message immediately. If True, send the message at the time specified by SendTime.
SendTime	Date	The date and time to send the message if DelaySend is True. The SendTime parameter must consist of a date and time string in US time format (m/d/y). Valid examples: 2/15/2007 12:10 PM 2/15/2007 14:10 2/15/2007 8:05 AM 2/15/2007 8:05 The time is interpreted as the time zone specified by the UserID used for the send. You may change your UserID time zone by visiting the TextPower customer web site at http://customer.TextPower.com. The time zone change is under the My Account | Change Password/TZ tab. **The user is responsible for knowing the location of the**

		caller and adjusting the time for the time zone that the recipient is in! **Warning: If no time is supplied after the date, the message will be sent at midnight!** If the SendTime is in the past, the message will be sent within the next minute. SendTime is accurate only to the nearest minute. Adding seconds to the time string is not an error but is ignored. If DelaySend is false, the SendTime parameter must still be set to a valid date but will be ignored.
Output Base Node:		SendToSendList
Sub Nodes		
Tag	**Level-Type**	**Meaning**
MemberCount	2-Integer	The number of members who are scheduled to receive this message.
SendTime	2-Date	The date and time of the scheduled send.
Output Example		

```
<SendToSendList xmlns="">
        <MemberCount>1</MemberCount>
        <SendTime>3/17/2008 1:04:46 PM</SendTime>
</SendToSendList>
```

Note about multiple sends: If the send list requested for sending is already in the process of executing, the specified send list and its members will be cloned into another send list. The cloned send list will then be executed. The cloned send list will be given a name of the form: *Client-OriginalSendList-mmddyy-n*. mm is the month, dd is the day, yy is the year and n is a sequence number. N will normally be 1 unless a third execution is requested. **Note that regardless of the original name, the cloned send list is marked as a temporary list that will eventually be automatically deleted!**

MergeSendList

Function MergeSendList(ListName As String, ListDB As DataSet, CreateList as Boolean) As XmlNode

Merge new numbers into an existing send list or optionally create a new list. If a number is already in the send list, the remaining items are updated. If it is not in the send list, the number is added.

Input Parameters		
Name	**Data Type**	**Meaning**
ListName	String	The name of the list to which the numbers in ListDB shoud be merged
ListDB	DataSet	A dataset containing the numbers that should be merged.
CreateList	Boolean	Create the list as new. If CreateList is True, a new list of the name ListName will be created. If CreateList is true and ListName already exists, an error will be returned.
Required Columns in ListDB		
CellNumber	String	The cell number that you are going to send to.
Carrier	String	The 5 digit carrier code for this number.
Optional Column in ListDB		
Comment	String	A 50 character or less comment field for this number.
Output Base Node:		SendToSendList
Sub Nodes		
Tag	**Level-Type**	**Meaning**
NumbersSupplied	2-Integer	The quantity of numbers supplied to be added or merged.
NumbersAdded	2-Integer	The quantity of numbers added to the Send List.
NumbersUpdated	2-Integer	The quantity of numbers updated in the Send List.
Output Example		
<SendListMerge xmlns=""> <NumbersSupplied>2</NumbersSupplied> <NumbersAdded>0</NumbersAdded> <NumbersUpdated>2</NumbersUpdated> </SendListMerge>		

DeleteSendList

Function DeleteSendList(ListName As String) As XmlNode
Delete the named send list and all its members.

Input Parameters		
Name	**Data Type**	**Meaning**
ListName	String	The name of the list to which the message should be sent.
Output Base Node:		SendListDelete
Sub Nodes		
Tag	**Level-Type**	**Meaning**
ListCount	2-Integer	The number of lists deleted. It should be 1.
MemberCount	2-Integer	The number of members that were deleted.
Output Example		
<SendListDelete xmlns=""> <ListCount>1</ListCount> <MemberCount>3</MemberCount> </SendListDelete>		

ManageSendTags

Function ManageSendTags(ByVal Command As Integer, ByVal CellNumber As String, ByVal TagList As String) As XmlNode

The **ManageSendTags** function is one primary way to add, delete or retrieve Send Tags for a particular cell number in a Client List. Please see our separate documentation on Send Tags at http://www.TextPower.com/Util/Docs/SendTags.html

A Client List is the list of all cell numbers Opted-In against a particular short code and keyword. The relevant short code and keyword can be determined from the credentials in the header and does not need to be re-entered in other parameters.

Input Parameters		
Name	**Data Type**	**Meaning**
Command	Integer	The command to be executed. Valid commands are 1 for setting tags onto a number, 2 for deleting tags from a number and

		3 for retrieving tags.
CellNumber	String	The 10 digit cell number found in the Client List pointed to by the credentials header.
TagList	String	A Tag List consists of a comma separated list of Send Tags. Send Tags are not case sensitive and are arbitrary alphanumeric strings of 75 characters or less and may not utilize the characters, ‘,<,>,” and of course a comma. See the above documentation for all the details.
Output Base Node:		ManageSendTags
Sub Nodes		
Tag	**Level-Type**	**Meaning**
Command	2-String	The name of the command corresponding to the numeric command on the input parameter. The valid command names are *Set Tag*, *Delete Tag* and *Get Tags*.
Result	2-String	The Result of the Tag Operation. Possible values are: **OK** – Tag operation successful **Partial** – Tag operation only partially successful **No Opt-In** – Tag operation failed because this cell number is not Opted-In.
TagCount	2-Integer	Number of tags for which processing was attempted if the command is 1 or 2. If the command is 3(Get Tags), the TagCount is the number of tags that are currently on the Cell Number passed as input.
SkippedTags	2-XML	The SkippedTags element contains an attribute called Count that counts the number of tags for which processing was skipped. If the count is greater than 0, Tag elements will be enclosed for each tag that was skipped. Common causes of skipped tags are adding a tag that is too long or has invalid characters in it. It is not an error to add a tag that already exists or to delete one that does not exist. ManageSendTags will not add a duplicate tag.
TagList	2-String	The TagList is a comma separated list of all the tags on the cell number after all processing has been completed.
Output Example 1: Successful add of two tags.		

```
<ManageSendTags xmlns=""><Command>Set Tag</Command>
<Result>OK</Result>
<TagCount>2</TagCount>
<SkippedTags count="0" />
<TagList>abc,xyz</TagList></ManageSendTags>
```

Output Example 2: Attempted add of three tags, one of which is invalid, TagList=" VIP, 568,75<34"

```
<ManageSendTags xmlns=""><Command>Set Tag</Command>
<Result>Partial</Result>
<TagCount>2</TagCount>
<SkippedTags count="1">
        <Tag item="0">75&lt;34</Tag>
</SkippedTags>
<TagList>abc,xyz</TagList></ManageSendTags>
```

Output Example 3: Simple Retrieval of existing tags.

```
<ManageSendTags xmlns=""><Command>Get Tags</Command>
<Result>OK</Result>
<TagCount>2</TagCount>
<TagList>abc,xyz</TagList></ManageSendTags>
```

Output Example 4: The Cell Number is not Opted-In

```
<ManageSendTags xmlns=""><Command>Set Tag</Command>
<Result>No Opt-In</Result>
<TagCount>1</TagCount>
<SkippedTags count="0" /></ManageSendTags>
```

GetSendTags

Function GetSendTags() As DataSet
The GetSendTags function retrieves a dataset with all the Send Tags installed on each number in the Client List. Just as with the ManageSendTags and GetClientList functions, the Client List is determined from the header information and does not need to be supplied on other parameters. Note that since you can have multiple Send Tags on a number, the CellNumber may repeat. The primary keys for this dataset are CellNumber and Tag.

Input Parameters(none)		
Columns Returned		
Column	**DataType**	**Meaning**
CellNumber	String	A cell number in the Client List
ShortCode	String	The Short Code that the Client List is attached to.
Keyword	String	The Keyword that the Client List is attached to.
Tag	String	A Send Tag attached to this Cell Number
AddDate	Date	The date and time that the tag was added.

DeleteSendListMembers

Function DeleteSendListMembers(ListName As String, ListDB As DataSet, DeleteAll As Boolean) As XmlNode

Input Parameters		
Name	**Data Type**	**Meaning**
ListName	String	The name of the list to which the numbers in ListDB shoud be merged
ListDB	DataSet	A dataset containing the numbers that should be merged.
DeleteAll	Boolean	Delete all members. A ListDB must be supplied if the DeleteAll parameter is true but is ignored.
Required Columns in ListDB		
CellNumber	String	The cell number that you are going to send to.
Output Base Node:		SendListMemberDelete
Sub Nodes		
Tag	**Level-Type**	**Meaning**
NumbersSupplied	2-Integer	The quantity of numbers found in ListDB
NumbersDeleted	2-Integer	The quantity of numbers deleted from the Send List. Note that if the DeleteApp option is used, NumbersDeleted can be greater than NumbersSupplied.
Output Example		

```
<SendListMemberDelete xmlns="">
        <NumbersSupplied>2</NumbersSupplied>
        <NumbersDeleted>2</NumbersDeleted>
</SendListMemberDelete>
```

MergeOptinList

```
Function MergeOptinList(ByVal ListDB As DataSet) As XmlNode
```

```
Add/Delete/Update numbers into an Optin List
```

The MergeOptinList function allows the user to update an Optin list on a periodic basis to keep it synchronized with your current records. MergeOptinList accepts up to 100 numbers at a time. Numbers can be marked for add, delete or update. Send Tags can be supplied as well. To keep the system responsive and avoid connection timeouts, the actual changes to the Optin list are performed in the background after conclusion of the MergeOptinList command. Most change requests sent via MergeOptinList will be effective within a few minutes. The user of the MergeOptinList command assumes responsibility under the TCPA regulations that all Optins have been consented to. The Optin List that the numbers supplied will operate on is determined by the basic credentials supplied.

NOTE: The MergeOptinList was added to the Advanced Services V3 package after its initial publication date. If you had an existing web service for the Advanced Services V3 package, you may need to refresh your web service to see this function.

Input Parameters		
Name	**Data Type**	**Meaning**
ListDB	DataSet	A dataset containing the numbers that should be merged. The data will be in the first table of the dataset, E.g. ListDB(0)
Required Columns in ListDB		
PhoneNumber	String	The cell number that you are going to send to.
Action	String	Use the letter A for add, D for delete and U for update. Update applies to both updating the carrier and updating the send tags,
Tags	String	The send tags to be placed on this number. The send tags are listed sequentially separated by commas. All existing send tags will be deleted and replaced by the tags in the Tags parameter.
Output Base Node:		MergeOptinList
Sub Nodes		
Tag	**Level-Type**	**Meaning**
NumbersSupplied	2-Integer	The quantity of numbers supplied to be added, deleted or updated.
NumbersAdded	2-Integer	The quantity of numbers added to the Optin List.
NumbersDeleted	2-Integer	The quantity of numbers deleted from the

		Optin List
NumbersUpdated	2-Integer	The quantity of numbers updated in the Optin List.
NumbersFailed	2-Integer	The number of failed numbers. Failed numbers are any that fail the form check of a North American number(10 digits, 2nd digit is a 0 or a 1). Don't put a 1 in front of numbers!
Output Example		
<MergeOptinList xmlns=""> <NumbersSupplied>2</NumbersSupplied> <NumbersAdded>0</NumbersAdded> <NumbersDeleted>0</ NumbersDeleted > <NumbersUpdated>2</NumbersUpdated> <NumbersFailed>1</ NumbersFailed > </ MergeOptinList >		

GetOptInCount

```
Function GetOptInCount() As XmlNode
Get Opt-in counts by keyword.
```

Input Parameters (none)		
Output Base Node:		TagCounts
Sub Nodes		
Tag	**Level-Type**	**Meaning**
sms	2-String	The count of members who are opted-in with a cellphone number.
email	2-String	The count of members who are opted-in with a e-mail address.
voice	2-String	The count of members who are opted-in with a landline number.
Output Example		

```
<TagCounts xmlns="">
        <sms>1</sms>
        <email>0</email>
        <voice>2</voice>
</TagCounts>
```

Note that this function call uses the credentials from the SOAP header to extract the necessary ShortCode and Keyword information. The ShortCode and Keyword are used to filter aggregated data.

GetTagCount

```
Function GetTagCount(ByVal TagList As String) As XmlNode
Get Opt-in counts by Tag list.
```

Input Parameters		
Name	**Data Type**	**Meaning**
TagList	String	Comma-delimited list of tags
Output Base Node:		GetTagCount
Sub Nodes		
Tag	**Level-Type**	**Meaning**
TagList	2-String	The list of tags used to extrapolate the opt-in counts.
TagCounts	2-Container	A container used to hold the number of opt-ins for each category: sms, email, and voice.
sms	3-String	The count of members who are opted-in with a cellphone number.
email	3-String	The count of members who are opted-in with a e-mail address.
voice	3-String	The count of members who are opted-in with a landline number.

Output Example

```
<GetTagCount xmlns="">
        <TagList>tag1,tag2</TagList>
        <TagCounts>
            <sms>1</sms>
            <email>0</email>
            <voice>2</voice>
        </TagCounts>
</GetTagCount>
```

Note that this function call uses the credentials from the SOAP header to extract the necessary ShortCode and Keyword information. The ShortCode and Keyword are used to filter aggregated data.

Errors Return

Note that an Errors Return can only occur on functions returning an XMLNode. Functions returning a DataSet can only return the DataSet or a SOAP Exception.

Input Parameters		None
Output Base Node:		Errors. The Count attribute has the number of errors contained in the block.
Sub Nodes		
Tag	**Level-Type**	**Meaning**
Error	2-Complex	Each error is described by an attribute in the Error tag. A Number, Type and Description attribute are supplied for each error.
Output Example 1-Invalid Subscription ID		
<Errors Count="1" xmlns="">		
<Error Number="30" Type="SubscriptionData" Description="Subscription ID 1 was not found as belonging to this account." />		
</Errors>		
Output Example 2-Invalid UserID		
<Errors Count="1" xmlns="">		
<Error Number="1" Type="User" Description="Invalid UserID" />		
</Errors>		

Soap Exception

Applications must handle the SOAP exception. Please report SOAP exceptions for non-structural errors along with the complete text of the exception to support@textpower.com. For functions that return a DataSet, credentials errors or structural errors will come in a SOAP exception. These types of exceptions do not indicate software faults with the TPI system.