TECHNICAL DOCUMENTATION
FOR:

BASIC SOAP/XML Interface Specifications Version 4.1

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

This manual describes Version 4.1 of the BasicXMLV4 SOAP interface to TextPower, Inc. (TPI) services. There are multiple levels of SOAP interfaces. Only the BasicXMLV4Interface is described in this manual.

Note that the name of this service, BasicXMLV4, has changed from that of previous versions. This new service provides two new features:

a. The ability to send text messages in nearly any language, including Asian languages such as Chinese and Korean. (**NOTE: This is an optional feature that will incur an additional cost if activated.)**
b. The ability to send text messages up to 1530 characters in length. These text messages are perceived as a single message by the sender (the TextPower customer) and by the recipient if they have a smart phone. Non-smart phone customers will still see multiple messages. This feature works only with cell carriers, not pagers. **(NOTE: Sending a message in this manner will be perceived by the sender and the recipient as a single message but the sender will be charged for each 160-character-or-less segment. A message of 350 characters, for example, will be charged as three [3] messages.)**

The language feature above requires a change to your account setup. If you wish to use this feature, please contact TextPower and request that your campaign setup be changed to accommodate it. These new features currently work on direct, queued, bulk and deferred sends. They are not available yet for Alert Dispatcher and Alert Manager. All of the above exceptions still handle the existing Extended length messages described later in this document.

If you do not wish to use the new features in this new API, you may still use it as your standard API for the standard operations supported in the previous API releases. New customers should start using this API. Existing customers do not need to change APIs unless they wish to start using these new features.

In addition, some other changes are of note. Some API calls in this new set have fewer parameters on them, mostly involving carrier information. In general, TextPower recommends that the carrier not be supplied on any API call as it will be automatically determined without user input.

As the name implies, this interface is an XML interface. There now is an interface called BasicJSONV4, which as its name implies is a JSON interface. Both the XML and JSON versions provide identical capabilities.

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

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.

All TPI BasicXMLV4 SOAP functions return an XML fragment or XML node. A SOAP header is required on all calls. Structural errors involving invalid credentials, incorrect parameter values, etc. return an XML fragment with errors as the base node. SOAP exceptions are supported. SOAP exceptions always represent internal program faults. Hopefully, you will not receive any but if you do, please forward the information to TPI support. The return of an Errors node represents a detectable error in the information supplied. Please check the information that you are supplying to correct this type of problem.

Changes in Version 4.1

New sending parameters on the SendSMS call to support multiple languages and long messages have been added. The method of handling API keys has been changed to make it simpler. The SendSMSFull command has been eliminated. Use SendSMS for all sends.

API Keys

An API key is a long, 128 bit number that is totally unique. An API Key looks like this: {E103D087-5299-47F5-A769-DBF4BC017B30}. You can get API Keys by going to the TextPower customer web site at https://customer.textpower.com and selecting the **Get API Key** menu item under the **Campaigns** tab. If you have more than one campaign, you must select the campaign you wish to get an API Key for first. To do this, select the Select menu item under the **Campaigns** tab first. On the API Key screen, just select the keyword and the UserID that you want a key for. Click the Add/Change Key button and you will be issued an API Key. Please note that if you return to the Get API Key screen and click this Add/Change API Key button again, you will be issued a new API Key which will invalidate your old key! If you change either your UserID or your password, your API Key will NOT change. If you have more than one UserID, each UserID will have a separate API Key.

WSDL and Header

The WSDL document for Version 4.1 of the basic XMLV4 SOAP interface is located at http://www.textpower.com/TPIServices/BasicInterfaceXMLV4.asmx?wsdl

The address for the SOAP calls on the basic XMLV4 interface is:
http://www.textpower.com/TPIServices/BasicInterfaceXMLV4.asmx

Alternatively, you may access the Basic XMLV4 Message Interface on a secure basis at:
WSDL
https://secure.textpower.com/TPIServices/BasicInterfaceXMLV4.asmx?wsdl
Address for SOAP Calls
https://secure.textpower.com/TPIServices/BasicInterfaceXMLV4.asmx

Note 1: TextPower has a high reliability server failover system. In order for this to work, users must use the www in front of textpower.com in URLs! Usage of the secure.textpower.com address already meets the requirements of the high reliability system.

Note that the interface points specified in previous versions of this document remain active.

If you use the SOAP header, it has four parameters on all calls as described in the WSDL. All header parameters are of type String.

Parameter	**Meaning**
UID	Required. The value of your user ID as assigned by TPI. If you are using an API Key, place it in this parameter and leave all other parameters blank.
PWD	Required. The value of your password as assigned by TextPower (this password can be changed at your discretion.). If you are using an API key, leave this parameter blank.
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. If you are using an API key, leave this parameter blank.
Keyword	The keyword in your campaign that you are using for this call. If you have only one keyword in your campaign, this parameter may be zero length string. If you are using an API key, leave this parameter blank.

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

Note that this SOAP interface does not support WAP messages. in raw form.

Extended Length Messages

The rules noted below for using Extended Length Messages only apply to messages sent to destinations outside the U.S (including Canada and the Caribbean). *Thanks to an advancement in TextPower's message processing, as of March 2019, extended length messages sent to U.S. destinations require no special coding, API parameters, handling or processing. It is fully automated so messages of any length can be sent. You will still be charged for each 160-character segment of a message, however, as that is the way carriers charge TextPower.*

Extended length messages are supported on the BasicXMLV4 interface are available on all sending methods via the new isMultiPartMsg parameter. Using this parameter, sends of 1530 characters are supported. On smart phones, these messages will appear as a single unified message.

Binary Message Sending

TextPower supports binary sending to sending to cell phones on short codes by special provisioning. To send binary you must:

1. Have a campaign specifically set up to send binary. A single campaign can send either normal text or binary but not both.
2. Comply with carrier regulations regarding certification so that specific carriers will allow your binary sends to go through. Please note that for some carriers, particularly T-Mobile and Verizon Wireless, gaining permission to send binary can be both time consuming and expensive.

TextPower supports sending binary content with or without UDH headers. When sending binary, your message must consist of only the hexadecimal character set plus an optional header separator. You may not send binary messages using the Queue option of SendSMS or send binary messages on a bulk send. It does not matter whether the hex characters are upper or lower-case.

Sending without a UDH

To send binary without a UDH, simply supply your message as hex text characters. E.g. 546ad3f478

Sending with a custom UDH

To send data with a custom UDH, simply supply your header first, a separation mark consisting of a double pipe symbol (||) followed by your text. E.g. the message `06 05 04` 0b8423f0||567b2a469e means send a UDH of 0b8423f0 and a message content of 567b2a469e.

Function Calls

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.
Note that in the WSDL, the Integer type is referred to as int, the Date type is referred to as DateTime and the XmlNode type is a complex type. 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 actual XML sent from the 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***! 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 all dates and times in the basic specification of both inputs and outputs are expressed in your local time.

Emails: You can also OptIn and send to Email addresses as well as Cell Numbers.

SendSMS

```
Function SendSMS(CellNumber As String, Message As String, PorceOptin As
Boolean, Queued As Boolean, DelaySend as Boolean, SendTime as DateTIme,
LanguageEncoding as Integer, IsMultiPartMesg AS Boolean) As XmlNode
```

Send an SMS Message using the full range of capabilities that TPI offers on this interface. In Version 4, this API call provides all the possible parameters for full operation on the TextPower system. There are no longer two versions of sending parameters. If your campaign is enabled for a US short code, a Toll Free Number or a TextEnabled Landline, you can use all features in this API call. If your campaign is set to use a Canadian short code, a Long Code which is not a Toll Free number or a TextEnabled landline, the LanguageEncoding parameter must be set to 0 and the IsMultiPartMsg parameter must be set to False since these routes do not have these extended features.

Input Parameters		
Name	**Data Type**	**Meaning**
CellNumber	String	The number of the cell phone to which the message is to be sent. The cell number must be exactly 10 digits for North American traffic. For international traffic, your number length limits will vary. When you sign up for international traffic, you must request specific length requirements. You may specify multiple numbers by simply separating them with a comma. For an immediate send, you may specify 10 numbers. For a queued or delayed send, you may specify up to 50 numbers.
Message	String	The message to be sent. The message must be at least 5 characters long. The length rules for version 4 are a bit more complex. For binary sends, the length can be much longer. For binary data sending, please see the section above. 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 unless you uses the international option described below. 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!
ForceOptin	Boolean	If your account has permission, you may force an OptIn for this account by setting this parameter to true. If a subscriber has done an OptOut however from an earlier OptIn, the ForceOptIn parameter has no effect.
DelaySend	Boolean	If the DelaySend Parameter is True, the message will be sent at a later time determined by the SendTime parameter.
SendTime	Date	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 that is set for this user in their User information. **Warning: If no time is supplied after the date, the message will be sent at midnight Pacific Time!** 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.
Queued	Boolean	If the Queue Parameter is true, the message is put on a queue for sending. This queue is serviced every 0.5 seconds so delivery time will not be noticeably different from a regular send. Use this parameter when you need a faster response for real time reasons. Certain high volume customers may be required to use this parameter. **All customers with the following exceptions are highly encouraged to use Queued=True!** Exceptions are: 1. Sends to the USA Mobility and American Messaging paging carriers 2. International Sends **3.** Sends using the unified International/North American feature where international sends can

		be simply done by prepending a + and a country code to the number.
LanguageEncoding	Integer	If 0, only the standard English characters are allowed. If 2, then all characters for all languages are accepted. For option 2, the message input from your system must be in Unicode.
IsMultiPartMsg	Boolean	**UPDATE (3/19)** Messages to any U.S./Domestic destination can be of any length. *No special coding or parameter handling is necessary* If IsMultiPartMsg is True, you may submit a message that is up to 1530 characters in length. You need not delimit it in any way. If the recipient has a smart phone, the SMS message will appear on their phone as one unitary message. Please note that "under the hood", the message is actually sent in 153 character chunks with linking codes attached to each message. Each segment will result in a message charge however. Thus, a message of 200 characters will be sent in two chunks and result in your standard message charge X 2.
Output Base Node:	<SMSImmediate>, <SMSDeferred>, or <SMSQueued>	
Sub Nodes		4.
Tag	**Level-Type**	**Meaning**
<Customer>	2-String	Customer name
<TimeStamp>	2-String	The time of these messages. TimeStamp is in your local time.
<MessageStatus>	2-String	The master tag for the following tags describing the message status
<SendResult SeqNum="n">	3-String	If there are no errors and sending was attempted, a SendResult tag will be presented for each number in the CellNumber parameter. The SeqNum tag enumerates the numbers. There will be one SendResult parameter for each number in the CellNumber parameter. If you used the IsMultiPartMsg parameter set to 2, only one SeqNum tag will appear, even though your message was split into multiple parts. The MultiPart splitting is conducted by the carrier and not by TextPower.

<MessageID>	4-Integer	A unique number for this message attempt. Immediate, queued and deferred sends all have different number ranges.
<CellNumber>	4-String	The cell number to which the message was sent (10 digits)
<Status>	4-String	*Sent* or *Not Sent* If Not Sent is the status for an immediate send. A colon will follow it with a reason for the not sent status if it was not sent. *Queued* or *Defered* will be returned for all successful delayed or queued messages. Status is not reported if Errors are sent.
Output Example 1(Immediate Send)		

```
<SMSImmediate xmlns="">
<Customer>@TPI Admin</Customer>
<MessageStatus>
    <TimeStamp>1/9/2017 10:41:31 PM</TimeStamp>
        <SendResult SeqNum="0">
            <MessageID>359071</MessageID>
           <CellNumber>3125550134</CellNumber>
           <Status>Sent</Status>
        </SendResult>
</MessageStatus>
</SMSImmediate>
```

Output Example 2-Queued Send)

```
<SMSQueued xmlns="">
<Customer>@TPI Admin</Customer>
<MessageStatus>
    <TimeStamp>1/9/2017 10:43:19 PM</TimeStamp>
    <SendResult SeqNum="0">
        <QueueID>20</QueueID>
        <CellNumber>3125550134</CellNumber>
        <Status>Queued</Status>
     </SendResult>
</MessageStatus>
</SMSQueued>
</SMSQueued>
```

Output Example 2- Double Number Send, one failing do to No OptIn

```
<SMSImmediate xmlns="">
<Customer>CUSTOMER A</Customer>
<MessageStatus>
```

```
        <TimeStamp>2/10/2008 8:49:27 PM</TimeStamp>
        <SendResult SeqNum="0">
                <MessageID>137017</MessageID>
                <CellNumber>312555788</CellNumber>
                <Status>Sent</Status>
        </SendResult>
        <SendResult SeqNum="1">
                <MessageID>0</MessageID>
                <CellNumber>6305558866</CellNumber>
                <Status>Not Sent:No OptIn</Status>
        </SendResult>
</MessageStatus>
</SMSImmediate>
```

Commentary on LanguageEncoding and IsMultiPartMsg Usage

The rules noted below for using Extended Length Messages only apply to messages sent to destinations outside the U.S (including Canada and the Caribbean). *Thanks to an advancement in TextPower's message processing, as of March 2019, extended length messages sent to U.S. destinations require no special coding, API parameters, handling or processing. It is fully automated so messages of any length can be sent. You will still be charged for each 160-character segment of a message, however, as that is the way carriers charge TextPower.*

If you specify the LanguageEncoding parameter to be zero and the IsMultiPartMsg parameter to be false, the operation of this API call is exactly like that of the SendSMS calls of previous versions.
If you specify LanguageEncoding to be 2 and IsMultiPartMsg to be true, you may send in almost any language on earth, including plain English, in lengths of up to 1530 characters. Please note that the characters in some languages such as Chinese, Japanese and Korean will count as two characters. So if your message is completely in Chinese characters or Japanese Kanji, your maximum will be 765 characters. If LanguageEncoding is set to 0 and you supply a foreign character, you will get an error. If IsMultiPartMsg is false and you supply a message of over 160 characters (135 for Canada), you will get an error.

It may be more helpful to think of these two parameters as restrictors rather than enablers. Set them to disable things that you don't want to do.

The standard TextPower Campaign setup allows you to use this API for all previously existing functions. Under the standard campaign setup, you will get an error if you try to set LanguageEncoding to something other than 0 or if you try to set the IsMultiPartMsg to True. If you wish to use the new features, contact TextPower and request that your campaign(s) be updated to allow these new features.

Note that the TextPower applications such as Alert Manager and Alert Dispatcher do not currently support multilingual sending and long, auto-split messages.

SendToClients

```
Function SendtoClients(ByVal msg As String, ByVal DelaySend As Boolean,
ByVal SendTime As Date, LanguageEncoding as Integer, isMultiPartMsg as
Boolean) As XmlNode
```

The SendToClients function sends an SMS to all SMS Opted In Clients as well as as Email to all Email Opted In Clients. The clients are the complete set of all Cell Numbers and Emails that are Opted In under the campaign and keyword referenced in the header.

Input Parameters		
Name	**Data Type**	**Meaning**
Msg	String	The message to be sent. This command supports extended length messages. See the Extended Length message section above.
DelaySend	Boolean	See SendSMS
SendTime	Date	See SendSMS
LanguageEncoding	Integer	If 0, only the standard English characters are allowed. If 2, then all characters for all languages are accepted. For option 2, the message input from your system must be in Unicode.
isMultiPartMsg	Boolean	If IsMultiPartMsg is True, you may submit a message that is up to 1530 characters in length. You need not delimit it in any way. If the recipient has a smart phone, the SMS message will appear on their phone as one unitary message. Please note that "under the hood", the message is actually sent in 153 character chunks with linking codes attached to each message. Each segment will result in a message charge. Thus, a message of 200 characters will be sent in two chunks and result in your standard message charge X 2 for each number.
Output Base Node:		
Sub Nodes		
Tag	**Level-Type**	**Meaning**
Type	2-String	The literal "Clients"
TimeFrame	2-String	The literals "Now" or "Deferred" depending upon the type of send requested
Count	2-Integer	The count of clients (SMS+Email) that the message will be sent to.
Output Example		

```
<GroupSend xmlns="">
<Type>Client</Type>
<TimeFrame>Now</TimeFrame>
<Count>9</Count>
</GroupSend>
```

SendToClientsByTag

```
Function SendtoClientsByTag(ByVal msg As String, ByVal TagList As
String, ByVal DelaySend As Boolean, ByVal SendTime As Date,
LanguageEncoding as Integer, isMultiPartMsg as Boolean) As XmlNode
```

The SendToClientsByTag function allows a client send to be tailored to only a defined sub-set of the total client list. This is achieved by using Send Tags. Please see our separate documentation on Send Tags at http://www.TextPower.com/Util/Docs/SendTags.html. The SendToClientsByTag operates exactly as does the SendToClients function but provides the one additional input parameter of TagList. A Tag List consists of a comma separated list of Send Tags that have been installed on your client list. Send Tags can be installed by using the ManageSendTag function which is found on the Advanced SOAP interface and by other means which are detailed in the documentation file listed above. Send Tags are not case sensitive and are arbitrary alphanumeric strings with some restrictions. The **TagList** parameter is limited to 600 characters. See the above documentation for all the details on Tag Lists.
Refer to the documentation on the SendToClients call for all other particulars of the SendToClientsByTag function. If Email Clients are defined, emails will also be sent to the Email Clients.

Examples:

TagList=”VIP” sends to only those clients who have the Send Tag, VIP set on them
TagList=”204,VIP” sends to only those clients who have either the VIP or the 204 Send Tag set on them.
TagList=”” sends to all clients and is exactly equivalent to a SendToClients call.

GetCounts

Return the count of all current OptIn Clients and of all subscription packages for this account

```
Function GetCounts() As XmlNode
```

Input Parameters		(None)
Output Base Node:		AccountStats
Sub Nodes		
Tag	**Level-Type**	**Meaning**
Clients	2-Integer	The count of all currently Opted In clients is in the Count attribute.
Output Example		
<AccountStats xmlns=""> <Clients Count="9" /> </AccountStats>		

GetAccountData

Get basic data on your TPI account.

```
Function GetAccountData() As XmlNode
```

Input Parameters		(None)
Output Base Node:		AccountDataionion
Sub Nodes		
Tag	**Level-Type**	**Meaning**
Account	2-Integer	Your internal account number
SendsLeft-3	2-Integer	If your account is on a pay as you go basis, this parameter gives the number of sends left in your account. If your account is not on a pay as you go basis, this number is -1.
ShortCode	2-String	The short code that this account is on.
OptInLevel	2-Integer	The OptIn level of your account. 0=No OptIn 1=Single OptIn 2=Double OptIn
TimeStamp	2-Date	The current server timestamp in Universal time
TimeAdjust	2-Integer	The number of hours to subtract from Universal time to get your local time. This assumes that you have set the Time Zone and Daylight Savings Time parameters for your account using the TextPower customer web site. If you have not set them, TimeAdjust will show the difference between Universal time and Pacific time.
LastReceiveID	2-Integer	The ID of the last received message. If no messages have been received, LastRecieveID=0
LastSendID	2-Integer	The ID of the last sent message. If no messages have been sent, LastSendID=0

Output Example

```
<AccountData xmlns="">
<Account>57</Account>
<SendsLeft>-1</SendsLeft>
<ShortCode>85700</ShortCode>
<OptInLevel>1</OptInLevel>
<TimeStamp>2/10/2009 5:00:30 PM</TimeStamp>
<TimeAdjust>5</ TimeAdjust >
<LastReceiveID>255837</LastReceiveID>
<LastSendID>2378160</LastSendID>
</AccountData>
```

LookupCarrier

Lookup carrier can be used to look up a cell phone carrier. You must have additional permission to execute this function. A charge applies for the use of this function.

```
Function LookupCarrier(PhoneNumber As String) As XmlNode
```

Input Parameters		
Name	**Data Type**	**Meaning**
PhoneNumber	String	The phone number that you wish to look up
Output Base Node:		CarrierData for.
Sub Nodes		
Tag	**Level-Type**	**Meaning**
PhoneNumber	2-String	It may have punctuation but must evaluate to a 10 digit US phone number
Result	2-String	The Result of the lookup. OK means the number was found as a cell phone and all the other parameters are valid.
CarrierCode	2-String	The TPI carrier code string. If the carrier code is INACP then that means that this number "Is Not A Cell Phone". Do not try to send to this number! Only failures will result! Lookups of fictitious movie numbers will receive this code and will be charged.
CarrierName	2-String	The common name for the carrier. Note that this carrier name comes from a national data base and the carrier name may not match the exact form that carriers require for marks purposes. If the carrier is not found, this parameter will be UNK.

Output Example 1: Phone is a cell phone

```
<CarrierData xmlns="">
<PhoneNumber>312555788</PhoneNumber>
<Result>OK</Result>
<CarrierCode>31003</CarrierCode>
<CarrierName>Verizon</CarrierName>
</CarrierData>
```

Output Example 2: Phone is not a cell phone

```
<CarrierData xmlns="">
<PhoneNumber>6304690000</PhoneNumber>
<Result>Not a Cell Phone</Result>
<CarrierCode>INACP</CarrierCode>
<CarrierName>UNK</CarrierName>
</CarrierData>
```

GetSMSStatus

GetSMSStatus returns the delivery status of a previously sent message. There are 3 ways of referring to the message that you want to query for. If you have a MessageID, use that. You got a MessageID when you did an immediate send earlier. If you have previously queried for status on this message using the Queued or Deferred IDs and the message had been sent, you received a MessageID back. Use that MessageID for all subsequent queries on that call.
If you have only a QueueID, use that parameter.
If you have only a DeferID, use that parameter.
Only one parameter may be non-zero. Both the other parameters must be zero! If you use QueueID or a DeferID and the message has not been sent, you will receive an Errors return.
GetSMSStatus can only be guaranteed to return status for messages sent in the previous 2 days. Fictitious movie number sends will always show no status since no send took place.

```
Function GetSMSStatus(MessageID As Integer, QueueID As Integer, DeferID
As Integer) As XmlNode

Function GetSMSStatus_Key(APIKey as String, MessageID As Integer,
QueueID As Integer, DeferID As Integer) As XmlNode
```

Input Parameters		
Name	**Data Type**	**Meaning**
MessageID	Integer	The MessageID if you sent the message originally without using either the Delay or Queued parameters.
QueueID	Integer	The QueueID if you sent the message originally using the Queue parameter
DeferID	Integer	The DeferID if you sent the message originally using the Delay parameters
Output Base Node:		MTStatus
Sub Nodes		
Tag	**Level-Type**	**Meaning**
MessageID	2-Integer	The MessageID which references this message. You may not have received a MessageID on your original send but all QueueIDs and DeferIDs will be converted to a MessaageID after the message has been sent. For QueueIDs, this usually occurs within a second or two For a DeferID, it will not occur until the message has been sent. Depending upon the parameters used to send the message, this could be months, days, hours, etc. in the future.

FirstStatus	2-String	The status returned from the carrier when the message was first sent. If this is not OK, then there will be no valid LastStatus.
LastStatus	2-String	The Last Status on this message. Messages may go through several status points before reaching a final status. Last Status is a word description of the status. NA means **Not Available**.
StatusDate	2-Date	The date and time that the after status was received.
Reason	2-Integer	A numeric code for the last status. 0 indicates no status.
Output Example 1- Successful retrieval		
<MTStatus xmlns=""> <MessageID>137001</MessageID> <FirstStatus>OK</FirstStatus> <LastStatus>Message successfully delivered.</LastStatus> <StatusDate>2/8/2008 1:05:17 PM</StatusDate> <Reason>4</Reason> </MTStatus>		
Output Example 2 –Query on a deferred message that has not yet been sent		
<Errors Count="1" xmlns=""> <Error Number="23" Type="MessageID" Description="Message ID 0 not found for this campaign." /> </Errors>		

Errors Return

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 a attributes 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 along with the complete text of the exception to support@textpower.com.

MO (Mobile Originating) service

MO service uses a combination of a short code number and keywords to route your message. Keywords are the first words at the beginning of the message. TextPower does a Post or Get to your application with the data for the call. POST/GETs can be performed for received calls, keyword help requests or OptOut requests. The basic interface is the same for each type of call. It is expected that different URLs will posted to for each type of function. There is no parameter supplied to distinguish between different call types. While technically, this is not a SOAP/Web Service interface, it is also published here to provide complete specifications for both the MO interface and the MT SOAP Basic interface in the same document.

.

POST and GET Interfaces

The Post and GET interfaces are identical in parameters. The only difference is in how the parameters are sent.

Parameters:

Parameters	**Meaning**
UID	The user ID that you provided us
PWD	The Password that you provided us
CellNumber	The originating number of the phone that sent the message.
Carrier	The carrier code of the phone that sent the message.
Message	The actual message. Not applicable on. The Message parameter contains the raw message received. The keyword may be on the front or not. It will not be there if this was a sticky reply or if a Block Keyword was found. The message will be text in ASCII for most customers. For certain kinds of accounts receiving binary, the information will be in ASCII encoded hexadecimal. If the original MO was a STOP message, the Message parameter will be "QuitQuit"
KW	The keyword that was used to route the message. Note that this keyword may not actually appear in the message. It may have been inferred due to a sticky reply or may be a Block Keyword.
ShortCode	The short code that applies to the call type.
StickyID	If this MO was routed to you via a normal Short Code/Keyword combination, StickyID is 0. If there was no keyword on the MO and an MT was sent to this phone in the last 30 minutes and you have sticky processing on your

	account, StickyID is the MT that this MO is responding to. The StickyID will be the same as that of a MessageID parameter for an SMSImmediate XML return.
ReceiveID	A unique integer reference number for this MO
OrigMessage	This parameter is only supplied when a STOP message is received. The Message parameter will be "QuitQuit" on all STOP calls. In case you need the original raw text which came in, it will be supplied on the OrigMessage parameter. Remember that this parameter is only supplied for STOP messages. Be prepared in your code for this parameter to not be present.
RouteHint	The RouteHint is non-zero only if StickyID is non-zero. If the RouteHint is non-zero, it a number unique to the user that sent the sticky message.
Tags	If SendTags are present on the CellNumber, the optional Tags parameter will appear. It will contain a comma separated list of the SendTags for this number. E.G. "tag1,tag2,tag3". Remember that this parameter is only supplied for numbers that have SendTags on them. Be prepared in your code for this parameter to not be present.

Each POST must have a response sent by the user. Please do not send a whole HTML page. A simple OK is sufficient. Note that the carrier is always sent to you on a POST or GET. If you will be responding to TextPower with a message send over Sender.aspx, please include the carrier you were sent in the initiating POST or GET. It will have a slight effect in lowering your message charges. It is recommended that when returning messages to TextPower after a POST, you use the queued option on the send. This will help in mitigating timeout errors. Even better is to implement queuing on your server also. If you separate the receiving process from the sending process, timeouts will be very rare.

Opt Out Interface

You may choose to have Opt Out notifications sent to your application. If you choose this option, your application is responsible for sending the required carrier compliant OptOut message to the user. The TextPower system has already performed its own OptOut action before you receive this notification. The interface is identical to the POST/GET interface. When an OptOut has occurred, the Message parameter always contains *QuitQuit*. This is the same regardless of the actual quit word entered by the user. No extra information from the MO is provided.

OptIn/OptOut Notifications

You may choose to have Optin and/or Optout events POSTed or emailed to your application. The data supplied is as follows:

Parameters	Meaning
ShortCode	The Short Code on which the optin or optout occurred
Keyword	The Keyword on which the optin or optout occurred
CellNumber	The CellNumber on which the optin or optout occurred
Action	The word OptIn or OptOut depending upon which action occurred.
Time	The Time at which the action occurred. The time is in the user's local time.
UID	The UID parameter which you have selected to be sent to you. If you have not selected this option, UID will be empty.
PWD	The PWD parameter which you have selected to be sent to you. If you have not selected this option, PWD will be empty.

Help Interface

You may choose to have Help requests sent to your application. If you choose this option, your application is responsible for sending the required Help message to the user. The interface is identical to the POST/GET interface.

Opt Out and Help messages may be sent to the same URL. The Opt Out/Help URL must be different from the MO delivery URL.