TextPower

Alert and authenticate using text messaging

TECHNICAL DOCUMENTATION FOR:

JSON Interface Specifications Version 4

CONFIDENTIAL

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

Interfaces Version 4

This manual describes the JSON Version 4.0 of the basic SOAP interface to TextPower, Inc. (TPI) services. There are multiple levels of JSON interfaces. Both the Basic Interface and the Web Management interfaces from the XML versions are described in this manual.

The name of this service is BasicJSONV4. The numbering is consistent, feature wise, with the XML interfaces. There are no JSON 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.
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.

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 and deferred sends. They are not available yet for Alert Dispatcher, Alert Manager and Bulk sends using the Client Members API calls. 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 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. MO means Message Originating. That is a message coming into a phone. The terms terminating and originating always are from the perspective of the phone.

Table of Contents

Interfaces Version 4

Overview

This manual will not attempt to describe the general JSON interface and structure. It is expected that the reader already knows this. This interface allows the TextPower (TPI) customer to build extremely sophisticated messaging applications

All TPI BasicJSONV4 JSON functions return a JSON object. Structural errors involving invalid credentials, incorrect parameter values, etc. return an JSON object with errors as the base node. The return of an Errors object 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 over Version 3 and earlier

New sending parameters on the SendSMS call to support multiple languages and long messages. 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 portal 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.

Address for JSON Calls https://secure.textpower.com/TPIServices/BasicJSONV4API

POST your JSON to the above address. There is no parameter name necessary.
Note that the interface points specified in previous versions of this document for XML remain active.

For the Version 4 JSON API calls, there is a common object element called msgcredentials. This consists of 4 elements that provide TextPower with your credentials information. All API calls must have this information filled in! All header parameters are of type String.

Interfaces Version 4

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.

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.*

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 generally take only 160 characters (135 for Canada) 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.

The Extended Length Messages described in this section are only for sends using the two functions above. For general sending, more extensive length options are available. They are described in other sections of this document.

Function Calls

The JSON APIs all include a **method** parameter as their first element. The **method** is the API function that you intend to call and is always mandatory. It is followed by the **msgcredentials** object needed for authentication. After that, each function has its own unique parameters.

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 JSON interface allows 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 JSON 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.

Note for JSON formats below:
Wherever the value of a parameter below is specified below as either true or false, the opposite is also available. The value is specified as the more common value.
Where numeric values are specified, there may be other values. Consult the table below for alternative values.
Quoted values must have real data substituted.

SendSMS

```
{
	"method": "SendSMS",
	"msgcredentials": {
		"UID": "uid",
		"PWD": "pwd",
		"Campaign": "campaignname",
		"Keyword": "keyword"
	},
	"CellNumber": "cellnumber",
	"Message": "message",
	"ForceOptIn": false,
	"DelaySend": false,
	"SendTime": "sendtime",
	"Queued": true,
	"LangEncoding": 0,
	"IsMultiPartMsg": false
}
```

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. 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 use 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.
LangEncoding	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 the message is over 160 characters, it will automatically be sent out. Each segment sent will still result in a message charge. Messages sent to destinations outside the U.S,. including those to Canada and the Caribbean, must still follow the existing processing rules below 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.

JSON Response:	{
	"SMSImmediate": {
	"Customer": "customer",
	"MessageStatus": {
	"TimeStamp": "1/31/2017 5:08:11 PM",
	"SendResult": {
	"@SeqNum": "int-num",
	"MessageID": "messageid",
	"CellNumber": "cellnumber",
	"Carrier": "carriercode",
	"TimeStamp": "1/31/2017 5:08:11 PM",	

	`"Status": "statuscode"`
	`}`
	`}`
	`}`
	`}`	

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	3-String	If there are no errors and sending was attempted, a SendResult object 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 set 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-String	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)
StatusCode	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.

Commentary on LanguageEncoding and IsMultiPartMsg Usage

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.

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 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

```
{
	"method": "SendToClients",
	"msgcredentials": {
		"UID": "uid",
		"PWD": "pwd",
		"Campaign": "campaignname",
		"Keyword": "keyword"
	},
	"Message": "message",
	"DelaySend": false,
	"SendTime": ""
}
```

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**
Message	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
Output Example		

{
"GroupSend": {
"Type": "Client",
"TimeFrame": "Now",
"Count": "19"
}
}		

TimeFrame	String	The word “Now” to indicate the send is immediate or Deferred
Count	String	The number of sends that will take place

SendToClientsByTag

```
{
      "method": "SendToClientsByTag",
      "msgcredentials": {
            "UID": "uid",
            "PWD": "pwd",
            "Campaign": "campaignname",
            "Keyword": "keyword"
      },
      "Message": "message",
      "TagList": "taglist",
      "DelaySend": "False",
      "SendTime": ""
}
```

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 both the VIP and 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 for this campaign and keyword

```
{
	"method": "GetCounts",
	"msgcredentials": {
		"UID": "uid",
		"PWD": "pwd",
		"Campaign": "campaignname",
		"Keyword": "keyword"
	}
}
```

Input Parameters		(None)
Output Base Node:		AccountStats
Sub Nodes		
Tag	**Level-Type**	**Meaning**
Clients	2-Integer	The count of all currently OptedIn clients is in the Count attribute.
Output Example		
`{"AccountStats":{"Clients":{"@Count":"count"}}}`		
`@Count" is the number of opted-in clients.`		

GetAccountData

Get basic data on your TPI account.

```
{
        "method": "GetAccountData",
        "msgcredentials": {
                "UID": "uid",
                "PWD": "pwd",
                "Campaign": "campaignname",
                "Keyword": "keyword"
        }
}
```

Input Parameters	(None)
Output Example:	AccountData
{"AccountData":{"Account":"account","SendsLeft":"numsends","ShortCode":"shortcode","OptInLevel":"optinlevel","TimeStamp":"1/31/2017 10:57:52 PM","TimeAdjust":"-8","LastReceiveID":"lastrecvid","LastSendID":"lastsendid"}}	

Parameter	**Meaning**
Account	Your internal account number
SendsLeft-3	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	The short code that this account is on.
OptInLevel	The OptIn level of your account. 0=No OptIn 1=Single OptI
TimeStamp	The current server timestamp in Universal time
TimeAdjust	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	The ID of the last received message. If no messages have been received,

	LastRecieveID=0
LastSendID	The ID of the last sent message. If no messages have been sent, LastSendID=0

LookupCarrier

Lookup carrier can be used to look up a cell phone carrier. A charge applies for the use of this function.

```
{
        "method": "GetCounts",
        "msgcredentials": {
                "UID": "uid",
                "PWD": "pwd",
                "Campaign": "campaignname",
                "Keyword": "keyword"
        },
        "PhoneNumber": "phonenumber"
}
```

Input Parameters		
Name	**Data Type**	**Meaning**
PhoneNumber	String	The phone number that you wish to look up
Outputs		CarrierData for.
Sub Nodes		
Tag	**Level-Type**	**Meaning**
PhoneNumber	String	It may have punctuation but must evaluate to a 10 digit US phone number
Result	String	The Result of the lookup. OK means the number was found as a cell phone and all the other parameters are valid.
CarrierCode	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	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:		

{
"CarrierData": {
"PhoneNumber": "phonenumber",
"Result": "OK",
"CarrierCode": "carriercode",
"CarrierName": "carriername"
}		

```
}
```

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.

```
{
      "method": "GetSMSStatus",
      "msgcredentials": {
            "UID": "uid",
            "PWD": "pwd",
            "Campaign": "campaignname",
            "Keyword": "keyword"
      },
      "MessageID": "msg-id-num",
      "QueuedID": "queue-id-num",
      "DeferID": "defer-id-num"
}
```

Input Parameters (Note: These parameters may be sent as strings or numbers.		
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	**Type**	**Meaning**
MessageID	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 MessageID 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	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	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	Date	The date and time that the after status was received.
Reason	Integer	A numeric code for the last status. 0 indicates no status.

Output Example 1- Successful retrieval

```
"MTStatus": {
                "MessageID": "137001",
                "FirstStatus": " OK ",
                "LastStatus": " Message successfully delivered ",
                "StatusDate": "2/8/2008 1:05:17 PM ",
                "Reason": "4"
        }
```

JSON Interface for Web Management Calls

SendPassCode

```
{
	"method": "SendPassCode",
	"msgcredentials": {
		"UID": "uid",
		"PWD": "pwd",
		"Campaign": "campaignname",
		"Keyword": "keyword"
	},
	"CellNumber": "cellnumber",
	"ServiceName": "servicename"
}
```

The SMS message sent to the user will be;
Your Pass Code for {ServiceName} is nnnn.
{ServiceName} will be replaced with what you sent to us in the Service Name parameter. Nnnn will be the Pass Code generated by TextPower. It will be either 4 or 5 digits. You do not need to send Pass Codes to Email subscribers.

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.
ServiceName	String	An acceptable name for your service under CTIA/MMA/Carrier rules.
Output Base Node:		PassCodeInfo
Sub Nodes		
Tag	**Level-Type**	**Meaning**
PassCode	String	The pass code generated and sent to the client
Carrier	String	The carrier code.
Output Example		

{
"PassCodeInfo": {
"PassCode": "passcode",
"Carrier": "carrier"
}
}		

ManageOptIn

```
{
	"method": "ManageOptIn",
	"msgcredentials": {
		"UID": "uid",
		"PWD": "pwd",
		"Campaign": "campaignname",
		"Keyword": "keyword"
	},
	"Action": "actionnumber",
	"CellNumber": "cellnumber"
}
```

Manage the OptIn function for a number when subscribing a customer from a web site. Note that if the campaign has the Force OptIn attribute set and the cell number has previously Opted Out for whatever reason, the Manage Optin call will fail(see example 5). The number can only be re-OptedIn via an MO optin.

Input Parameters		
Name	**Data Type**	**Meaning**
Action	String	The OptIn action to be performed. Allowable case insensitive values are: Set- create an OptIn Reset – remove an OptIn Query – Get the current OptIn state
CellNumber	String	The number of the cell phone or email address that is to be acted upon for the OptIn action. A cell number must be exactly 10 digits for North American traffic. An email address must be well formed.
Output Base Node:		OptIn
Sub Nodes		
Tag	**Level-Type**	**Meaning**
TimeStamp	Date	The date of the information retrieval or action
Action	String	The action that was performed
CellNumber/Email	String	The cell number or email address on which the action was performed. If the action was performed on a Cell Number, the tag is named CellNumber. If the action was performed on an Email Address, the tag is named Email.
Result	String	The result of the requested action. Possible results: For command set:

		• Campaign is not marked for OptIn • OptIn Created • Cannot Force Re-Optin For command reset: • Optin not active: No Action • OptIn Deleted For command query • No OptIn • Waiting for Yes • Opted In Error-(code)

Output Example 1

```
{
	"OptIn": {
		"TimeStamp": "1/31/2017 10:57:52 PM",
		"Action": "query",
		"CellNumber": "6345550144",
		"Result": " No OptIn "
	}
}
```

Output Example 2a: Create a Cell Number Optin

```
{
	"OptIn": {
		"TimeStamp": "1/31/2017 10:57:52 PM",
		"Action": "set",
		"CellNumber": "6345550144",
		"Result": " OptIn Created "
	}
}
```

Output Example 2b: Create an Email Address Optin

```
{
	"OptIn": {
		"TimeStamp": "1/31/2017 10:57:52 PM",
		"Action": "set",
		"CellNumber": "myEmail@MyEmail.com ",
		"Result": " OptIn Created "
	}
}
```

SetOptinTimeout

```
{
	"method": "SetOptInTimeout",
	"msgcredentials": {
		"UID": "uid",
		"PWD": "pwd",
		"Campaign": "campaignname",
		"Keyword": "keyword"
	},
	"CellNumber": "cellnumber",
	"Minutes": "minutes"
}
```

The SetOptinTimeout function can be used when you have a need to automatically release an OptIn after a certain period of time. This can arise when using Virtual Dedicated Short Codes when a free dialog is needed but only for a limited period of time. You could also effect the release by using the ManageOptIn function and using your own timer. For users who cannot run timer functions on their own servers however, the SetOptInTimeout function is very useful. SetOptinTimeout will only work on an existing Optin which must have been previously created with an MO or a ManageOptIn call. When the OptIn is released, no message will be sent to the Cell Number that is being released.

Input Parameters		
Name	**Data Type**	**Meaning**
CellNumber	String	The number of the cell phone that is to have its OptIn released. The cell number must be exactly 10 digits for North American traffic.
Minutes	Integer	The number of minutes into the future that the OptIn is to be released. The timing of the OptIn release is not precise. It will never be less than the minutes specified but can be as many as 9 minutes later.
Output Example:		OptIn

```
{
	"OptIn": {
		"TimeStamp": "1/31/2017 10:57:52 PM",
		"Action": "action",
		"CellNumber": "cellnumber",
		"TimeOutAt": "1/31/2017 11:00:52 PM"
	}
}
```

Tag	**Level-Type**	**Meaning**
TimeStamp	Date	The date of the action in Universal Time
Action	String	The action that was performed. The

		successful Action is **TimeOut Set.**
CellNumber	String	The Cell phone number that the Time Out was set on. The Short Code and Keyword of this cell phone number are derived from the credentials.
TimeOutAt	String	The time in Universal Time that the Optin will be released. The timing of the OptIn release is not precise. It will never be less than the time listed but can be as many as 9 minutes later.

SetAutoWelcome

```
{
      "method": "SetAutoWelcome",
      "msgcredentials": {
            "UID": "uid",
            "PWD": "pwd",
            "Campaign": "campaignname",
            "Keyword": "keyword"
      },
      "Message": "message",
      "MOLimit": "molimit",
      "Clear": "false"
}
```

This function sets the parameters on the TextPower AutoWelcome feature for the short code and keyword attached to the credentials you provided. In an MO based registration, you may set up your account to issue the carrier mandated welcome message upon the initial OptIn of a particular cell phone to the short code/keyword combination. This primitive allows the user to control their auto welcome. Once set via this primitive, you may use the information to send a welcome MT using the SendAutoWelcome primitive described elsewhere in this document or you may use this primitive to set the information for normal MO based OptIns.

Input Parameters		
Name	**Data Type**	**Meaning**
Message	String	The carrier compliant welcome message that will be sent to subscribers when they first OptIn. This message needs to be set only once for each Short Code/Keyword combination.

MOLimit	Integer	The limit for MOs that will be accepted for each cell number. This parameter is usually used to limit entries allowed for contests. A value of zero means no limit.
Clear	Boolean	If True, the Auto Welcome feature will be cleared. If False, it will be set.
Output Base Node:		OptIn
Sub Nodes		
Tag	**Level-Type**	**Meaning**
TimeStamp	Date	The date of the action
Action	String	The action that was performed
Result	String	The result of the requested action. Possible results: • Set • Reset

Output Example 1

```
{
        "OptIn": {
                "TimeStamp": "1/31/2017 10:57:52 PM",
                "Action": "AutoWelcome",
                "Result": "Set"
        }
}
```

Output Example 2

```
{
        "OptIn": {
                "TimeStamp": "1/31/2017 10:57:52 PM",
                "Action": "AutoWelcome",
                "Result": "Reset"
        }
}
```

SendAutoWelcome

```
{
        "method": "SendAutoWelcome",
        "msgcredentials": {
                "UID": "uid",
                "PWD": "pwd",
                "Campaign": "campaignname",
                "Keyword": "keyword"
        },
        "CellNumber": "cellnumber"
}
```

This function sends the Auto Welcome message to the listed cell number. The carrier need not be supplied as it will be recovered from the OptIn information. Attempting to send an AutoWelcome message to a cell number not opted in for the campaign and keyword in the credentials will result in a failure.

Input Parameters		
Name	**Data Type**	**Meaning**
CellNumber	String	The number of the cell phone that is to have the message sent to it. The cell number must be exactly 10 digits for North American traffic.
Output Base Node:		OptIn
Sub Nodes		
Tag	**Level-Type**	**Meaning**
TimeStamp	Date	The date of the action
Action	String	The action that was performed. Always **AutoWelcome Send**.
CellNumber	String	The Cell phone number that the message was sent to.
Result	String	The result of the requested action. Possible results: • Sent • NotSent
Output Example 1		

```
{
        "OptIn": {
                "TimeStamp": "1/31/2017 10:57:52 PM",
                "Action": "AutoWelcome Send",
                "CellNumber": "cellnumber",
                "Result": "result"
        }
}
```

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	Complex	Each error is described by 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",
                "Error": {
                        "@Number": "1",
                        "@Type": "User",
                        "@Description": " Invalid UserID "
                }
        }
}
```

Web Management JSON interfaces

OptIn Interface for web sites

When short codes and keywords require OptIn, some campaigns may be certified for using the web as an OptIn method. Using the web requires that the user enter their cell phone number and have displayed to them the same OptIn A message that they would see on their phone. You may customize this message for the web by eliminating things that apply only to cell phones. An example might be changing a statement such as "*Reply Yes to Accept*". After this, they may be considered opted in and messages may be sent to them according to the terms of the OptIn A message.

TextPower keeps the OptIn information in its own database. External applications where TextPower acts only as a gateway, need a way to create web based OptIns. That is what this interface does. OptIn now supports both Cell Numbers for SMS sending and Emails for Email Sending.

Process Flow for Web Based OptIn:

Cell Numbers

1. User is directed to the applicable web page.
2. User enters cell phone number.
3. User has displayed to them the terms of the sign up.
4. User is sent a special pass code to the phone that they wish to OptIn on.
5. User enters the pass code received on their phone into the web page. If the entered pass code and the transmitted pass code are the same, the user is Opted In. The pass code makes sure that the web user is in physical possession of the cell phone that is to be Opted In and prevents fraud or massive unauthorized OptIns of users.
6. User is given acknowledgement of the result of the transaction via a carrier compliant text message (Welcome Message).

Email Addresses

1. User is directed to the applicable web page.
2. User enters email address.
3. User has displayed to them the terms of the sign up.
4. User is sent a special email to the email address that they wish to OptIn on.
5. User clicks on the confirm link in the email to complete the OptIn.

Division of Responsibility

The application will be responsible for taking care of steps 1, 2 and 6 above. Steps 2, 4 and 5 require access to the TextPower OptIn database and must be taken care of by TextPower by calling functions detailed in this document.
The application interface is housed on a web page of the application's choosing that meets the interface standards set forth later in this document.

Application Design

Since direct access to the OptIn database is restricted to TextPower code, the user will need these primitives to access the database. There are two methods to accomplish this.

Method 1

House the OptIn code on TextPower's servers as in the example. Custom OptIn pages can be created for customers although there will be a charge for this. Customers often want their own domain name to show in the browser address bar however. You potentially could host your registration page on your own server with your own domain and put the Actual OptIn code in a frame that references a TextPower domain. Due to cross-domain security issues, there are very few browsers anymore that will permit the necessary references across domains. If you wish to take this route, you should be advised that the browser address bar would have to show a TextPower domain as in the example.

Method 2

As an application, you may host an OptIn page on your own domain and provide all the necessary logic there. To gain the access you need to create the actual OptIn, you will need to use the primitives supplied by TextPower. You must use the SOAP calls described in this document.

OptIn Logic Flow

The CTIA approved method for doing web OptIns for cell numbers is summarized below. This discussion does not apply to email addresses. You must follow this method to have your application certified by the carriers.

1. Collect the Cell Phone Number from the user on your web page. The carrier associated with the cell phone number will always be looked up automatically by TextPower. There is no need to collect it on the web site.
2. Display the terms such as charges and messages that they may receive to the customer. They must click on a button that indicates their agreement to those terms.

3. Send a Pass Code to the phone attempting to be registered. This is necessary to verify that the person on the web page is actually in possession of the phone that is being registered.
4. Have the person enter on your web page the pass code that they received.
5. If the pass code matches, you may Opt them in.

Of course, you should handle all the error conditions that can result from this process.

You will need to use these three primitives from TextPower to accomplish these tasks. The interfaces for these primitives are documented herein.

1. The ManageOptin call allows you to set, reset and query the state of a user’s optin to your keywords and short code.
2. The SendPassCode call allows you to send a pass code to a user. You cannot use TextPower’s regular sending primitives for this function since they will check the OptIn state before allowing a send. By definition, you don’t have an OptIn at this point.

Send the required, carrier compliant welcome message to the new subscriber. This may be done after OptIn using the user’s own logic or you may use the AutoWelcome feature of the TextPower system to send this message. This is controlled by the SetAutoWelcome and SendAutoWelcome primitives described in this document.

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 have identical 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.