



Internode Pty Ltd  
ABN 82 052 008 581  
PO Box 284 Rundle Mall, South Australia 5000  
Telephone 13 NODE Facsimile 1300 FXNODE  
[www.internode.on.net](http://www.internode.on.net)

## NodeText® Gateway User Guide

### Introduction

This document describes the HTTPS interface between **NodeText Gateway** clients and the Internode SMS Gateway. In particular, details are provided on the communication methods and data formatting requirements, between client systems and the Internode SMS Gateway HTTPS Interface.

*Please note that the NodeText service can be used to send text messages to any Australian mobile phone service, ie any mobile phone service starting with '04', even when roaming overseas. However, the service is **not** intended to support sending text messages to international mobile phone services.*

### Interface Overview

The Internode SMS Gateway HTTPS Interface (hereinafter referred to as "the Interface") provides a fast and secure link between NodeText Gateway clients and the Internode SMS Gateway.

A custom HTTPS client is required to successfully connect to and send SMS messages via the Interface. Such a client must conform to the specifications outlined in this document. If you require assistance to develop an HTTPS client for this purpose, please contact Internode technical services.

The Interface is located at: [https://txt.on.net/cgi-bin/sms\\_tcp.cgi](https://txt.on.net/cgi-bin/sms_tcp.cgi)

### Interface Operation

The Interface receives request parameters via a secure HTTP request, processes the request, and returns a detailed response. For specific information regarding the format of the request and response, please refer to the sections 'Interface Inputs' and 'Interface Outputs', respectively.

Once the request has been received, it is parsed. If any required parameters are not supplied, or any parameters are invalid, an error message is returned and no further processing is carried out. No SMS messages will be queued for sending if any part of the request is deemed invalid.

Should all input be validated and the request be approved, the message(s) will be prepared prior to queuing. Preparation involves the validation of recipients and the splitting of message text into multiple segments if required.

For requests with a large number of recipients and/or message segments, the order in which the individual messages are queued for sending becomes relevant. The SMS Gateway will attempt to queue messages to small subsets of the total recipient list in the order in which recipients were supplied as a parameter of the request. If multiple segments are to be sent, all segments will be queued for a single subset of recipients before proceeding to the next subset.

During the queuing process for a request with multiple segments per recipient, should the current segment fail to be sent to an individual recipient, the SMS Gateway will not attempt to queue any subsequent segments to that recipient for the current request. The failed segment and all subsequent segments will be reported as not sent by the Interface.

## **Input Method**

Communication with the Interface must be via a secure HTTP request to the interface URL specified above. Several parameters are required, which may be specified via the GET or POST request methods.

Please note: the URL of a http request (including any parameters passed via query string using the GET method) is limited to a maximum of 2,048 characters. For this reason, use of the POST method is strongly recommended. POST parameters are passed within the body of the http request, which may be of unlimited size, and are not appended to the limited-length URL as a query string (as per the GET method).

Each request is sent as a single line of information.

## **Input Parameters**

All request parameters must be supplied to the Interface via the query string (GET method) or request body (POST method), in key=value pairs separated by the & character.

Each parameter key should not be specified more than once, and parameter values must conform to the requirements outlined below. All keys and values are case sensitive. Key-value pairs may be specified in any order.

Please Note: Parameter values must be formatted correctly to ensure all characters are correctly interpreted by the Interface, refer to the 'Input Parameter Encoding' section for details.

The following parameters are accepted by the Interface:

<b>Key</b>	<b>Mandatory / Optional</b>	<b>Value</b>	<b>Example</b>
user	Mandatory	Your Internode SMS Service account username and realm.	user=user@host
pass	Mandatory	Your Internode SMS Service account password.	pass=myPass
dest	Mandatory	A comma separated list of fully qualified destination Australian mobile telephone numbers.  These must be in international calling format, without the leading international calling prefix (ie without 0011).  Hence, all destination numbers must consist of 11 digits beginning with 614.  A maximum of 50 recipients may be specified per request.	dest=61412345678,61487654321
src	Optional	A single fully qualified 'alias' Australian mobile telephone number, in international calling format as described above. This will appear to the recipient to be the sender of the message.  This alias must be registered with Internode.	src=61400123456

Key	Mandatory / Optional	Value	Example
msg	Mandatory	<p>A textual message to be sent to the destination number(s). Should the message exceed 160 characters in length (the maximum permitted by the SMS standard), it will be split into multiple segments (see the split parameter below) each segment to be sent individually, and charged as one message per segment per recipient.</p> <p>Refer to 'Input Parameter Encoding' for details of characters permitted in the textual message.</p>	<p>msg=Test%20message</p> <p>msg=The%20quick%20brown%20fox%20jumped%20over%20the%20lazy%20dog%20at%20midnight%20on%20February%2029th%20of%20a%20non%20leap%20year%20using%20Gregorian%20date%20forms%2E%20Find%20the%20deliberate%20mistake%20in%20this%20message%20of%20over%20160%20characters%2E</p> <p>Note the encoding used in the above message.</p>
split	Optional	<p>The method by which messages longer than 160 characters are split into multiple segments. Valid values are "hard" or "soft".</p> <p>The hard method splits messages into segments at the limit of message capacity, while the soft method splits message into segments on white space, and adds a prefix ".." and suffix "(Cont)" between segments.</p> <p>The default value is soft.</p>	<p><b>split=hard</b></p> <p>The above long message will be split into two segments :</p> <p><i>The quick brown fox jumped over the lazy dog at midnight on February 29th of a non-leap year using Gregorian date forms. Find the deliberate mistake in this me</i></p> <p>and :</p> <p><i>ssage of over 160 characters.</i></p> <p><b>split=soft</b></p> <p>The above long message will be split into two segments:</p> <p><i>The quick brown fox jumped over the lazy dog at midnight on February 29th of a non-leap year using Gregorian date forms. Find the deliberate mistake in (Cont)</i></p> <p>and</p> <p><i>.. this message of over 160 characters.</i></p>
timezone	Optional	<p>The timezone to which the request will be localised. The response timestamp(s) and any message logs provided to you will reflect this timezone.</p> <p>The default is Australia/Adelaide.</p>	<p>timezone=Australia%2FMelbourne</p> <p>timezone=Australia%2FSydney</p> <p>Note the encoding used in the above examples.</p>

Key	Mandatory / Optional	Value	Example
test	Optional	<p>A Boolean value used to prevent requests sent to the Interface from actually being sent.</p> <p>Set this value to "true" or "1" to test your HTTPS client before it is placed into production.</p> <p>The Interface will continue to evaluate requests whilst in test mode, and all output will be formatted in a manner that closely emulates that of normal operation.</p>	<p>test=true</p> <p>test=false</p>

### **Input Parameter Encoding**

Request parameter values must be formatted correctly for transmission, to ensure all characters are correctly interpreted by the Interface.

The Interface will accept 7-bit ASCII characters in the range 10, 32-126 only.

Certain non-alphanumeric characters must be converted to hexadecimal in accordance with RFC2396.

Refer to Appendix A: ASCII to Hexadecimal Conversion Chart for a complete list of valid ASCII characters and their hexadecimal representations.

Note - do not encode parameter keys or the special delimiter characters used to construct the parameter string (the characters ? = and &).

### **Input Examples**

The request examples below utilise the GET method (to illustrate parameters passed via the Query String), and are sent as a single line each.

#### ***Example 1 - One recipient.***

[https://txt.on.net/cgi-bin/sms\\_tcp.cgi?user=user@host&pass=myPass&dest=61412345678&msg=Test%20Message](https://txt.on.net/cgi-bin/sms_tcp.cgi?user=user@host&pass=myPass&dest=61412345678&msg=Test%20Message)

#### ***Example 2 - Two recipients, source specified.***

[https://txt.on.net/cgi-bin/sms\\_tcp.cgi?user=user@host&pass=myPass&dest=61412345678,61409876543&src=61401234987&msg=Test%20Message](https://txt.on.net/cgi-bin/sms_tcp.cgi?user=user@host&pass=myPass&dest=61412345678,61409876543&src=61401234987&msg=Test%20Message)

#### ***Example 3 - One recipient, split method specified, test mode specified.***

[https://txt.on.net/cgi-bin/sms\\_tcp.cgi?user=user@host&pass=myPass&dest=61412345678&msg=Test%20Message&split=hard&test=true](https://txt.on.net/cgi-bin/sms_tcp.cgi?user=user@host&pass=myPass&dest=61412345678&msg=Test%20Message&split=hard&test=true)

#### ***Example 4 - One recipient, timezone specified.***

[https://txt.on.net/cgi-bin/sms\\_tcp.cgi?user=user@host&pass=myPass&dest=61412345678&msg=Test%20Message&timezone=Australia%2FSydney](https://txt.on.net/cgi-bin/sms_tcp.cgi?user=user@host&pass=myPass&dest=61412345678&msg=Test%20Message&timezone=Australia%2FSydney)

## Output Format

Having received and processed a request from an HTTPS client, the Interface will return a detailed response to describe the result of the request.

For specific information regarding Interface request processing and failure condition handling, please refer to 'Interface Operation'.

The response contains five sets of information, formatted as follows:

### 1. Status Line

The status line comprises a numerical status code and textual status description of the result. It is returned as the first line of output under all circumstances.

The status line is formatted as:

Status: <status-code> <status-description>

Examples:

Status: 00 Total Success

Status: 10 Input Error (Invalid Syntax): destination +123 is invalid

Status: 20 Authentication Failure (Invalid Credentials): user, pass or source ip is invalid

Valid status codes are:

Code	Description	Meaning
00	Total Success	All messages queued for sending.
01	Partial Success	Some messages queued for sending.
02	Total failure	All messages failed to queue for sending.
03-09	Reserved	
10	Input Error (Invalid Syntax): [detailed description]	A required input parameter has been omitted, or an input value does not meet syntax constraints. Please Note: only ONE error is described. Thus if there are 3 errors on the request, only the first one encountered is returned.
11	Input Error (Invalid Protocol): https is required	A client has attempted to connect via the HTTP protocol instead of the HTTPS protocol.
12-19	Reserved	
20	Authentication Failure (Invalid Credentials): user, pass or source ip is invalid	Invalid credentials for your service. An invalid user, pass or source ip has been specified by the client.
21	Authentication Failure (Invalid Destination): destination(s) [dest] not permitted	Invalid destination for your service. The specified destination(s) have not been approved for your service.
22	Authentication Failure (Invalid Source): sender [sender] not permitted	Invalid source for your service. The specified sender has not been approved for your service.
23-29	Reserved	
30	System Unavailable	The system is currently unavailable for use by clients. Please contact Internode for details.
31-49	Reserved	
50	Unspecified Error	An unspecified error has occurred. The matter will be addressed by Internode as soon as possible.
51-99	Reserved	

### **1. Status Line (*continued*)**

The status description value provides useful information pertaining to the request status, and any failures that may have occurred.

The line is terminated by a newline character.

### **2. Successes Line**

The Successes line comprises a numerical value indicating the total number of SMS messages successfully queued for delivery. This value factors in multiple recipients and multiple segments.

The Successes line is returned for Status codes 00-09 inclusive.

The Success line is formatted as:

Successes: <success-count>

Example:

Successes: 12

The line is terminated by a newline character.

### **3. Failures Line**

The failures line comprises a numerical value indicating the total number of SMS messages that failed to be queued for delivery. This value factors in multiple recipients and multiple segments.

The Failures line is returned for Status codes 00-09 inclusive.

The Failures line is formatted as:

Failures: <failure-count>

Example:

Failures: 1

The line is terminated by a newline character.

#### 4. Message Segment Line(s)

The Message Segment line comprises a textual value indicating the text of SMS message queued for sending. If message splitting has occurred to produce multiple message segments (multiple SMS messages per recipient) from the supplied message text, individual segments will be reported on successive Message Segment lines.

The Message Segment line is returned for Status codes 00-09 inclusive.

The Message Segment line is formatted as:

Message: <Message segment>

Example original message:

*The quick brown fox jumped over the lazy dog at midnight on February 29th of a non-leap year using Gregorian date forms. Find the deliberate mistake in this message of over 160 characters.*

Assuming a soft split (the default), this message is split into two segments:

*The quick brown fox jumped over the lazy dog at midnight on February 29th of a non-leap year using Gregorian date forms. Find the deliberate mistake in (Cont)*

and

*.. this message of over 160 characters.*

Then two message segment lines are returned:

Message: The quick brown fox jumped over the lazy dog at midnight on February 29th of a non-leap year using Gregorian date forms. Find the deliberate mistake in (Cont)

Message: .. this message of over 160 characters.

Each line is terminated by a newline character.

## 5. Transmission Status Line(s)

The Transmission Status line comprises one or more numerical or textual values separated by white space, indicating the transmission status for each segment attempted to be sent for a single recipient. The status values are listed in segment number order.

Valid values for receipt are:

Value	Meaning
Number	The segment is queued for sending, and the host system has received a unique message id. This number may be recorded as a billing reference.
MSG-SENT	The segment is queued for sending, but no unique message id is available at this time.
MSG-NOT-SENT	The segment failed to queue for sending (no further details are available at this time).
MSG-NOT-SENT-TEST	The segment was not sent as the test flag was set to true.

If multiple recipients have been supplied, status values for individual recipients will be reported on successive message receipt lines.

The Transmission Status line is returned for Status codes 00-09 inclusive.

The Transmission Status line is formatted as:

<recipient telephone number>: {<number|MSG-SENT|MSG-NOT-SENT|MSG-NOT-SENT-TEST>}..

The status values are returned in segment number order.

Example original message:

*The quick brown fox jumped over the lazy dog at midnight on February 29th of a non-leap year using Gregorian date forms. Find the deliberate mistake in this message of over 160 characters.*

Split into two segments:

*The quick brown fox jumped over the lazy dog at midnight on February 29th of a non-leap year using Gregorian date forms. Find the deliberate mistake in (Cont)*

and

*.. this message of over 160 characters.*

If the message is sent to one recipient (61412345678) and split into two segments, both of which are successfully queued, but a reference number is only available for the first segment, then the following Transmission Status line is returned:

61412345678: 12340987 MSG-SENT

Each line is terminated by a newline character.

## **Output Examples**

### ***Example 1 - Total success: one recipient, one message segment, all messages queued for sending.***

Status: 00 Total Success  
Successes: 1  
Failures: 0  
Message: This is a test message.  
61412345678: 12340987

### ***Example 2 - Partial success: two recipients, two message segments, three messages queued for sending, one message failed to queue for sending.***

Status: 01 Partial Success  
Successes: 3  
Failures: 1  
Message: Message segment one... [truncated] (Cont)  
Message: .. Message segment two.  
61412345678: 12340988 MSG-NOT-SENT  
61409876543: 12340990 12340991

### ***Example 3 - Total Failure: one recipient, two message segments, all messages failed to queue for sending (preceding segment failed, subsequent segment(s) not attempted).***

Status: 02 Total Failure  
Successes: 0  
Failures: 2  
Message: Message segment one... [truncated] (Cont)  
Message: .. Message segment two.  
61412345678: MSG-NOT-SENT MSG-NOT-SENT

### ***Example 4 - Input error (Invalid Syntax).***

Status: 10 Input Error (Invalid Syntax): destination +123 is invalid

### ***Example 5 - Input error (Invalid Syntax).***

Status: 10 Input Error (Invalid Syntax): required parameter msg not specified

### ***Example 6 - Authentication Failure (Invalid Credentials).***

Status: 20 Authentication Failure (Invalid Credentials): user, pass or source ip is invalid

## **Full Communication Examples**

The following examples are of the total interface communication for a number of scenarios.

Note the request examples below utilise the GET method (to illustrate parameters passed via the Query String), and are sent as a single line each.

### ***Example 1 – One Recipient, Short Message, All successful.***

Message is "Test message" to be sent to 0412345678

*Request:*

[https://txt.on.net/cgi-bin/sms\\_tcp.cgi?user=user@host&pass=myPass  
&dest=61412345678&msg=Test%20Message](https://txt.on.net/cgi-bin/sms_tcp.cgi?user=user@host&pass=myPass&dest=61412345678&msg=Test%20Message)

*Response:*

Status: 00 Total Success  
Successes: 1  
Failures: 0  
Message: Test message  
61412345678: 12340987

### ***Example 2 - Two recipients, two message segments, source specified, partial success: three messages queued for sending, one message failed to queue for sending.***

Message is "The quick brown fox jumped over the lazy dog at midnight on February 29th of a non-leap year using Gregorian date forms. Find the deliberate mistake in this message of over 160 characters" to be sent to 0412345678 and 0409876543

*Request:*

[https://txt.on.net/cgi-bin/sms\\_tcp.cgi?user=user@host&pass=myPass  
&dest=61412345678,61409876543&src=61412345678  
&msg=The%20quick%20brown%20fox%20jumped%20over%20the%20lazy%20dog%20at%20midnight%20on%20February%2029th%20of%20a%20non%20leap%20year%20using%20Gregorian%20date%20forms%2E%20Find%20the%20deliberate%20mistake%20in%20this%20message%20of%20over%20160%20characters%2E](https://txt.on.net/cgi-bin/sms_tcp.cgi?user=user@host&pass=myPass&dest=61412345678,61409876543&src=61412345678&msg=The%20quick%20brown%20fox%20jumped%20over%20the%20lazy%20dog%20at%20midnight%20on%20February%2029th%20of%20a%20non%20leap%20year%20using%20Gregorian%20date%20forms%2E%20Find%20the%20deliberate%20mistake%20in%20this%20message%20of%20over%20160%20characters%2E)

*Response:*

Status: 01 Partial Success  
Successes: 3  
Failures: 1  
Message: The quick brown fox jumped over the lazy dog at midnight on February 29th of a non-leap year using Gregorian date forms. Find the deliberate mistake in (Cont)  
Message: .. this message of over 160 characters.  
61412345678: 12340988 MSG-NOT-SENT  
61409876543: 12340990 12340991

## Appendices

Samples of PERL code using the above features are available upon request.

### Appendix A : Parameter String

The Parameter String is appended to a URL as a query string (for GET requests), or contained within a request body (for POST requests); it conveys parametric data to the Interface.

The basic syntax is `parameter1=value1&parameter2=value2...`

If formatted as a Query String, the Parameter String is appended to the URL and begins with a question mark (?), which acts as a delimiter between itself and the URL.

The Parameter String consists of a series of `parameter=value` pairs, separated by the ampersand (&) character. Each parameter is separated from its corresponding value by an equal sign. For example:  
`msg=Hello&name=Bob&age=27`

Many special characters (including, but not limited to whitespace, ampersand and equal to) affect the function of the Parameter String. To preserve their integrity, these special characters must be encoded to hexadecimal representation.

Refer to Appendix B : ASCII to Hexadecimal Conversion Chart for a complete list of special characters in conventional, decimal and hexadecimal representations.

## Appendix B : ASCII to Hexadecimal Conversion Chart

The following characters are permitted by the Interface, and do not require conversion to hexadecimal format:

Character	Hexadecimal	ASCII Code
0-9	48-57	%30-%39
A-Z	65-90	%41-%5A
a-z	97-122	%61-%7A

The following characters are permitted by the Interface, but should be converted to hexadecimal format:

Character	Hexadecimal	ASCII Code
Newline	%0A	10
Space	%20	32
!	%21	33
"	%22	34
#	%23	35
\$	%24	36
%	%25	37
&	%26	38
'	%27	39
(	%28	40
)	%29	41
*	%2A	42
+	%2B	43
,	%2C	44
-	%2D	45
.	%2E	46
/	%2F	47
:	%3A	58
;	%3B	59
<	%3C	60
=	%3D	61
>	%3E	62
?	%3F	63
@	%40	64
_	%5F	95
~	%60	96

The following characters are not permitted by the Interface:

Character
[
\
]
^
{
}
~