“The Smart SMS Platform For Growing Businesses. Reliability Guaranteed”

Here’s why you should be using directSMS for all your messaging needs

You will get the help you need from an expert outfit with thousands of satisfied customers across Australia

You will be up and running in no time

“directSMS' customer service team have provided a high level of service and have been very responsive with all our support queries.”

Eliot Harper
Production Workflow
Marketing Manager
Fuji Xerox Australia

“directSMS customer service is very efficient. Whenever I have had to ask questions, I always got quick replies and they were very helpful”

Greg Gubiani
Service Express Manager
The Westin Melbourne

directSMS: PHP API

PHP API Reference Manual

Outline of directSMS’ Business Grade PHP API.

August 2014

Table of Contents

Introduction

Thank you for considering directSMS for your messaging needs. This document provides a reference manual for all features available via the PHP based HTTP interface to our API server.

The PHP API allows you to SMS enable your PHP based web applications and systems by integrating them with directSMS API servers over the HTTP protocol.

HTTPS is also supported for secure transactions using SSL encryption over the HTTP protocol. The PHP API allows for the sending of requests over plain HTTP as well as SSL for greater security. In order to make use of SSL, you will need to ensure your PHP installation is compiled with OpenSSL support.

Features and functionality

The PHP API servers will allow you to perform the following:

1. Send 1-way and 2-way SMS messages.
2. Schedule the sending of SMS messages for future delivery.
3. Retrieve 2-way SMS replies and correlate each reply with its original message.
4. Retrieve Inbound SMS messages if you have your own dedicated inbound number.

Application Development

The PHP API is distributed as PHP source code (***sms_connection.php***). It can be installed anywhere that your PHP interpreter has read access, and can access the internet.

The main class to be use is ***sms_connection***. This class is used to connect to the gateway and perform any of the operations outlined above.

In the event that an error is encountered while interacting with the SMS gateway, the error message describing the error condition can be retrieved by calling the ***get_error()*** method on the ***sms_connection*** class.

Access to the API servers comes with Australia based phone and email support.

Authenticating

In order to connect and authenticate to the server, call the ***connect()*** method passing your directSMS username and password. If the credentials passed are genuine, the method will return **true**. If the gateway

is down, the username and password are invalid in any way, or any one of a number of errors, the method will return **false**.

If the user account used is not set up for API access, an error will be returned. In order to use the API, you have to contact the directSMS Support team to enable this feature on your account. This will activate your 60 day API trial.

If you have an enterprise license key, the API will require you to pass the license key along with the login credentials for authentication when opening a new connection. This is the approach to take if you are developing an application that will be distributed to multiple customers with different directSMS accounts.

```php
<?php

    // Load the sms_connection class
    require_once("sms_connection.php");

    // Create a new connection, using SSL
    $conn = new sms_connection(true);

    // Check if there are any problems. At this stage
    // the only problem at this stage might be that
    // the SMS gateway is unreachable
    if($conn->is_error())
    {
        print("ERROR: " . $conn->get_error() . "\n");
    }
    else
    {
        // Login and start a session, passing the username, password
        if($conn->connect("myuser", "secret!"))
        {
            // go about your business
        }
```

Sending an SMS

Once you've called the connect operation, you can send 1-way or 2-way SMS messages using the ***send_one_way_sms()*** or ***send_two_way_sms()*** methods.

These methods accept an optional parameter **max_segments** which tells the SMS Gateway the upper limit to use when sending long SMS messages. Long SMS messages are those that exceed the standard 160 characters.

Long SMS Support

A single SMS can only carry 160 characters. By default, if you attempt to send a message which is more than 160 characters, the server will return an error.

You can explicitly indicate that you would like a long message to be split into multiple SMS segments if it exceeds this 160 character limit by passing the optional parameter **max_segments**.

This parameter indicates the maximum number of SMS messages the server should use while sending out messages longer than 160 characters.

Each message segment is restricted to 153 characters in length, with 7 bytes used by the gateway for headers so the destination handset can piece the message segments back together again once they have been received.

Each message segment is equal to sending a single SMS.

Please note; a message will only be split into multiple segments if it is longer than 160 characters. If your message is under this limit and you specify a **max_segments** value, the message will not be split and you will only be charged for a single SMS.

max_segments	Max. SMS Messages Sent	Characters Available
1	1	160
2	2	305 (153 x 2)
3	3	459 (153 x 3)
4	4	612 (153 x 4)
.	.	.
N	N	153 x N

Upon successful completion, both of these methods will return the unique identifier of the message submitted within directSMS' systems.

```
// Create a new message
$message  = "This is a test message from the directSMS gateway";

// Create an array of the mobile numbers to send this message to,
// mind the limit of 100 mobiles at any one time
$mobiles  = array("0401001001", "0402002002");

// Send a 1-way message with the sender id  "test 123"
$id = $conn->send_one_way_sms($message, $mobiles, "test 123");

print("1-way message id = " . $id . "\n");

// Send a 2-way SMS message with the message id "my-id". We can
// use that to search for replies to this message later
$id = $conn->send_two_way_sms($message, $mobiles, "my-id");

print("2-way message id = " . $id  . "\n");
```

In the event that you are sending long SMS (**$max_segments** > 1), these methods will return an ***array*** containing the 1 or more IDs of the individual message segments that make up the long SMS submitted.

If any errors come up, ***null*** will be returned instead and the methods ***is_error()*** and ***get_error()*** can be used to check for error encountered.

Unicode Support

Unicode messages are fully supported. This means you can send messages containing any 16-bit characters like Chinese, Korean, Hindi, Hebrew, Arabic, Russian and so on.

If you are sending Unicode messages, you just need to pass ***true*** as the optional 4th ***unicode*** argument into the ***send_one_way_sms()*** or ***send_two_way_sms()*** methods.

The ***sms_connection*** class will then convert the message text passed these methods into **UCS-2** encoded text before submitting the message to the directSMS servers.

Due to the increased character size, Unicode messages can only have a maximum of **70 Unicode characters**. If you need to send longer messages, you can make use of the ***max_segments*** parameter to have our systems break up the long message into multiple segments.

Please note that Unicode message segments can only hold **67 Unicode characters**. The rest of the message payload contains a User Data Header (UDH) containing message segmentation information. This header is used by the destination handset to put the original message together based on all the received segments.

Another note; your Unicode message will only be split into multiple segments if it is longer than 70 characters. If your message is under this limit specified by the ***max_segments*** value, the message will not be split and you will only be charged for a single SMS.

max_segments	Max. SMS Messages Sent	Unicode Characters Available
1	1	70
2	2	134 (67 x 2)
3	3	201 (67 x 3)
4	4	268 (67 x 4)
.	.	.
N	N	67 x N

Please see http://www.Unicode.org/ for more information.

Request Limitations

Please note, the PHP API makes HTTP GET requests back to the directSMS server. As such, there is a limit of 100 mobile numbers per call to the ***send_one_way_sms()*** or ***send_two_way_sms()*** methods.

Scheduling an SMS

Once authenticated and connected, you can schedule 1-way or 2-way message for delivery at a later date/time using the ***schedule_one_way_sms()*** and ***schedule_two_way_sms()*** methods respectively.

The date/time you wish the SMS to be sent at has to be passed in as the number of seconds after the Unix Epoch: January 1 1970 00:00:00 (GMT).

The scheduled send time will be rounded to the nearest time the send SMS daemon is scheduled to run which is typically every 10 minutes.

```
// Schedule a message to go out at 11 AM on Jan 1
// of next year to say happy new year
$time = mktime(11, 0, 0, 1, 1, date("Y") + 1);
```

```
// The message
$message = "Happy new year from the team @ directSMS team";

// The mobile numbers to send the message to
$mobiles = array("0401001001", "0402002002");

// Send to the gateway
$id = $conn->schedule_branded_sms($message, $mobiles,
                                  "directSMS", $time);

if($conn->is_error())
{
    print("Error scheduling new 1-way message = " .
          $conn->get_error() . "\n");
}
else
{
    print("Scheduled message id = $id\n");
}
```

The ***schedule_one_way_sms()*** and ***schedule_two_way_sms()*** methods return an identifier which uniquely identifies the submitted SMS message in directSMS' systems.

Should any error conditions arise while communicating with the gateway, ***null*** will be returned instead, and the methods ***is_error()*** and ***get_error()*** used can be used to check the error.

Request Limitations

Please note, the PHP API makes HTTP GET requests back to the directSMS server. As such, there is a limit of 100 mobile numbers per call to the ***schedule_one_way_sms()*** or ***schedule_two_way_sms()*** methods.

Cancelling Scheduled SMS

The gateway allows you to cancel any SMS messages that are scheduled for future delivery.

This can be very helpful for appointment confirmation type and similar applications where you schedule a reminder message but then need to cancel it once the appointment is rescheduled.

In order to cancel a scheduled message you need to specify the ID returned by the gateway when the message was first created.

Please note, only the user account used to create the original scheduled message is allowed to cancel it.

```
// Cancel the schedule message created earlier
// The 32 byte ID is stored in $id
$result = $conn->cancel_scheduled_sms($id);

if($conn->is_error())
{
    print("Error in cancelling scheduled $id message = " .
          $conn->get_error() . "\n");
}
else
{
```

```
            print("Scheduled message id = $id has been cancelled\n");
        }
```

The ***cancel_schedule_sms()*** method takes the identifier which uniquely identifies the scheduled SMS message in directSMS' systems. This is the same identifier returned by the ***schedule_one_way_sms()*** and ***schedule_two_way_sms()*** methods.

Should any error conditions arise while communicating with the gateway, ***null*** will be returned instead, and the methods ***is_error()*** and ***get_error()*** used can be used to check the error.

Receiving Delivery Receipts

The gateway can send a HTTP GET/POST in real time to your webserver whenever a Delivery Receipt (DR) is processed for one of the messages you sent earlier. Delivery receipts are short messages sent from the networks denoting the successful or otherwise delivery of an SMS message.

You can turn this notification mechanism on and control the URL the gateway will push the DR notification messages to by updating the User Profile page on the directSMS Customer Portal. You will receive each DR notification in real time in a separate HTTP request.

The parameters in the HTTP GET/POST that are sent to your server are as follows:

Parameter	Description
id	This is the identifier of the original message that this DR is correlated to within directSMS' systems e.g. 6df259577165fd4c7fa6ba0cc8fa41d5
type	The type of the original sent message e.g. 1-way or 2-way
mobile	The mobile number message was delivered to. This will be presented in International format e.g. 61412345678 instead of 0412 345 678
status	The status of the message in question. The different values are as follows: **DELIVRD** – Message was delivered successfully **UNDELIV** – Message was not delivered **EXPIRED** – The gateway attempted to send the message for 48 hours but was not able to deliver it because the handset was turned off or not in coverage

	REJECTD - The message was rejected by the network and was not delivered. This may happen if your messages are deemed as SPAM or if the subscriber does not exist. This depends on the destination network and whether they run a overt vs. covert filtering scheme **DELETED** - The message was aborted by the network because the it is a duplicate or the subscriber is invalid
when	The number of seconds since the message identified in the DR was received. For example, if the message was received by the handset in question 10 seconds ago, this value will be 10

Example

1. On the User Profile page, you have selected to send a HTTP push to the URL http://www.mywebsite.com/process_delivery_receipt?security_option=xyz when a new delivery receipt is received.
2. One of your 1-way messages has been delivered to its destination 10 seconds earlier.
3. The gateway will make the following HTTP GET call to your server: http://www.mywebsite.com/process_delivery_receipt?security_option=xyz&id=6df259577165fd4c7fa6ba0cc8fa41d5&type=1-way&mobile=61444123123&status=DELIVRD&when=10

Error Handling

If for any reason your server does not respond with a **200 OK** HTTP response code, the HTTP Push will be deemed a failure and the gateway will retry to push this DR notification again at a later time.

The gateway retries failed notifications every 20 minutes for between 1 and 2 hours. After this, the HTTP push is aborted for the given DR.

Getting SMS replies

You can receive reply messages to 2-way SMS messages sent out earlier in 3 ways:

1. Email Push

The gateway can send you an email in real time when a new reply message is received. The email will contain the original 2-way message, as well as the reply that was just received. To turn this notification mechanism on, you need to update your User Profile by logging onto directSMS' Customer Portal using your username and password.

Please note, the email will be sent to the email address you have nominated under your user profile.

2. HTTP Push

The gateway can send a HTTP GET/POST in real time to your server whenever a reply message is received. You can turn this notification mechanism on and control the URL the gateway will push the messages to by updating the User Profile page on the directSMS Customer Portal. This is the recommended way to receive reply messages. You will receive each reply in real time in a separate HTTP request.

The parameters in the HTTP GET/POST that are sent to your server are as follows:

Parameter	Description
id	This is the identifier of the original 2-way message that this reply is correlated to within directSMS' systems e.g. 6df259577165fd4c7fa6ba0cc8fa41d5
message_text	The content of the SMS message. This will be URL encoded
mobile	The mobile number of the person replying. This will be presented in International format e.g. 61412345678 instead of 0412 345 678
when	The number of seconds since this SMS was received. For example, if the message was received by directSMS' gateway 2 seconds ago, this value will be 2
messageid	This is the message identifier of the original 2-way SMS that identifies that message in your system, if one was supplied when the original 2-way SMS was sent out

Example

4. On the User Profile page, you have selected to send a HTTP push to the URL http://www.mywebsite.com/process_reply_sms?security_option=xyz when a new 2-way reply message is received.

5. One of your customers has replied to your message from earlier today to confirm their appointment at 3PM with the message “Yes I will be there”. The original message is identified as “APP1234” in your system.

6. The gateway will make the following HTTP GET call to your server: http://www.mywebsite.com/process_sms?security_option=xyz&id=6df259577165fd4c7fa6ba0cc8fa41d5&message_text=Yes+I+will+be+there&mobile=61444123123&messageid=APP1234&when=0

Error Handling

If for any reason your server does not respond with a **200 OK** HTTP response code, the HTTP Push will be deemed a failure and the gateway will retry to push this message again at a later time.

The gateway retries failed messages at 20 minute intervals for between 4 and 8 hours. After this, the HTTP push is aborted for that message.

3. Polling API Server

The ***get_reply_sms()*** method allows client applications to retrieve **unread** replies to **ALL** 2-way SMS messages or to a specific 2-way message.

The ***get_reply_sms()*** method takes an optional **messageid** parameter. If a message Id is specified, only **unread** replies to the referenced 2-way message will be retrieved.

```
// Retrieve any replies to the 2-way SMS submitted just now
$replies = $conn->get_reply_sms(true, "my-id");

print(count($replies) . " replies retrieved\n");

// Display the replies
for($i = 0; $i < count($replies); $i++)
{
    print($replies[$i]->to_string() . "\n");
}
```

The above code sample will result in the retrieval of all **unread** replies to the 2-way SMS message with the message ID “my-id” which was submitted earlier.

The ***get_sms_replies()*** method also takes a boolean parameter to mark any retrieved replies as **read**. This stops the server from retrieving these messages again in future calls to the ***get_reply_sms()*** method.

PLEAE NOTE: Polling for received messages excessively will see your account suspended. The most efficient solution is to use the HTTP Push feature where reply messages are pushed to your server as they are received in real time.

Getting Inbound SMS

If you have a dedicated inbound SMS number, you can retrieve your inbound SMS messages in three ways.

1. Email Push

The gateway can send you an email in real time when the message is received. You can control the email address the gateway will forward messages to through the Inbound SMS settings page on the directSMS Customer Portal.

2. HTTP Push

The gateway can send a HTTP GET/POST in real time when the message is received to your server. You can control the URL the gateway will push the message to through the Inbound SMS settings page on the directSMS Customer Portal.

This is the recommended way to receive messages. You will receive them in real time.

The parameters in the HTTP GET/POST that are sent to your server are as follows:

Parameter	Description
inbound_number	This will be the inbound number you are leasing from directSMS, this will represented in International format e.g. 61429007007
id	This is the identifier of the inbound message within directSMS' systems e.g. 6df259577165fd4c7fa6ba0cc8fa41d5
message_text	The content of the SMS message. This will be URL encoded
mobile	The mobile number of the SMS message sender. This will be presented in International format e.g. 61412345678 instead of 0412 345 678
when	The number of seconds since this SMS was received. For example, if the message was received by directSMS' gateway 2 seconds ago, this value will be 2

Example

7. On the Inbound SMS page, you have selected to send a HTTP push to the URL http://www.mywebsite.com/process_sms?security_option=xyz when a new inbound message is received.

8. A customer has decided to contact you via SMS on your inbound number 0428 001 001. The customer sends the SMS "This is John Citizen, please provide more info".

9. The gateway will make the following HTTP GET call to your server: http://www.mywebsite.com/process_sms?security_option=xyz&id=6df259577165fd4c7fa6ba0cc8fa41d5&inbound_number=61428001001&message_text=This+is+John+Citizen,+please+provide+more+info&mobile=61444123123&when=0

Error Handling

If for any reason your server does not respond with a **200 OK** HTTP response code, the HTTP Push will be deemed a failure and the gateway will retry to push this message again at a later time.

The gateway retries failed messages at 20 minute intervals for between 4 and 8 hours.

3. Polling API Server

You can poll the API server periodically looking for new inbound SMS messages. This is the least recommended of the three methods due to the load it places on both client and server.

The ***get_inbound_sms()*** method allows client applications to retrieve **unread** inbound SMS messages. The ***get_inbound_sms ()*** method takes an optional **inbound_number** parameter. If an inbound number is specified, only the **unread** messages received on the specified number will be retrieved.

```
// Retrieve any inbound SMS messages received and mark them as read
$messages = $conn->get_inbound_sms(true)
print(count($messages) . " inbound messages retrieved\n");

// Display the messages
for($i = 0; $i < count($messages); $i++)
{
    print($messages [$i]->to_string() . "\n");
}
```

The above code sample will result in the retrieval of all **unread** inbound SMS messages received on any of the inbound numbers leased by the current customer.

The ***get_inbound_sms ()*** method also takes a **boolean** parameter to mark any retrieved messages as **read**. This stops the server from retrieving these messages again in future calls to the ***get_inbound_sms()*** method.

PLEAE NOTE: Polling for received messages excessively will see your account suspended. The most efficient solution is to use the HTTP Push feature where messages are pushed to your server in as they are received in real time.

Getting your balance

The ***get balance()*** method can be used to retrieve the number of SMS credits left on the your directSMS account.

```
// Get the current balance
$credits = $conn->get_balance();

// Use the is_error() method to check for problems
if($conn->is_error())
{
    print("ERROR: " . $conn->get_error() . "\n");
}
else
{
    print("Current credit balance = " . $credits . " credit(s)\n");
}
```

Disconnecting

Once the client is finished sending and/or receiving all messages, they disconnect from the gateway by calling the ***disconnect()*** method.

How to get started

There are four simple steps to get started:

Step 1: Register online through our website. The system will give you some free credits to use to trial our services.

Step 2: Log in using the username/password sent to you.

Step 3: Email our support team to enable API access on your new account. Once activated, your account will have API access so you can test your solution.

Step 4: Start integrating your application, website or system.

If you require any more information or help, please call us on 1300 724 387 or email support@directsms.com.au and we'll be happy to assist.

If you are a software vendor, speak to us about our Partner Program or about becoming a Whitelabel Reseller. Please see details below.

Support Team
directSMS Pty Ltd
Email: support@directsms.com.au
Phone: 1300 724 387
Fax: 02 8569 0306
Web: www.directsms.com.au

Partner Program

The directSMS Partner Program is aimed at software vendors who wish to add SMS enabled features to their software offerings.

For each customer you introduce to directSMS through this Partner program, you will receive a generous trailing commission based on the revenue generated by that customer for the lifetime of their account.

As well as offering your customers the SMS features they need in your software, you can have the confidence that they will be backed by local-based technical and customer support. Not to mention a 100% money back reliability guarantee.

Best of all, directSMS will pay you a commission on each transaction as well as take care of all your clients' SMS related customer service and billing. What are partners for at the end of the day?

Contact us for more information.

Whitelabel Reseller

The whitelabel solution is aimed at software vendors who wish to SMS enable their applications while generating revenue from their clients' SMS services. Add profits to your software as an SMS services reseller.

Brand your own reseller sub-domain with your own logo and corporate colours. Your clients will not know that directSMS is providing the underlying service. We will be your perfect silent partner.

As a whitelabel reseller, you can resell using the method that is most convenient for your organisation:

1. You can avoid all billing and paperwork headaches by having our systems bill your clients on your behalf. All you do is set your own prices and collect the margin you set above our low wholesale rates.

2. Alternatively, you can go "full service" where we bill you for all your clients' usage each month. You can then bill your clients any way you see fit.

Ultimately, the choice is yours.

Contact us for more information.

Appendix A - Class Reference

sms_connection

This is the main class providing connectivity with the directSMS API server. The public methods are detailed below:

`sms_connection()`

This constructor takes a boolean argument *secure* which indicates whether or not to use SSL to communicate with the SMS gateway. Please note; In order to use SSL encryption, PHP 4.3 or higher is required.

`connect()`

This method will return a boolean after connecting and authenticating the user's login credentials against the directSMS gateway.

If the user account used is not set up for API access, a value of false is returned, along with an error message.

In order to use the API, you either need to contact the directSMS Support team to enable this feature on your account.

`is_error()`

This method will return a boolean value reflecting whether or not the last operation carried out using this sms_connection encountered an error condition.

`get_error()`

This method will return the error message associated with the error encountered during the last operation. If in fact an error condition is encountered. If the operation completed successfully, null is returned instead.

`send_one_way_sms()`

This method will send the *message* passed in with the *senderid* specified through the directSMS gateway. Upon success, this method returns the 32 byte Message ID used by the directSMS gateway to uniquely identify the submitted SMS message. In the case of failure, null is returned instead.

If you send a long SMS (where **$max_segments** > 1), the method will return an ***array*** containing the ID of each message segment making up the long SMS. If an error is encountered, null is returned.

If you wish to send Unicode (16-bit character) messages, you will need to pass true as the value of the ***$unicode*** parameter. Please note, the ***$message*** is expected to contain **UCS-2** characters.

Due to HTTP request size limitations, there is a limit of 100 mobile numbers for each message submitted.

`send_two_way_sms()`

This method will send the *message* passed in through the directSMS gateway, associating it with the *messageid* specified. This *messageid* can be used in later calls to ***get_reply_sms()*** to retrieve replies to this

message and this message only. On success, this method returns the 32 byte ID used by the directSMS gateway to uniquely identify the submitted SMS message. If the operation fails, null is returned instead.

If you send a long SMS (where **$max_segments** > 1), the method will return an ***array*** containing the ID of each message segment making up the long SMS. If an error is encountered, null is returned.

If you wish to send Unicode (16-bit character) messages, you will need to pass true as the value of the ***$unicode*** parameter. Please note, the ***$message*** is expected to contain **UCS-2** characters.

Due to HTTP request size limitations, there is a limit of 100 mobile numbers for each message submitted.

`schedule_one_way_sms()`

This method will save the *message* passed in with the *senderid* specified to be sent out at the date/time specified. Upon success, this method returns the 32 byte Message ID used by the directSMS gateway to uniquely identify the submitted SMS message. In the case of failure, null is returned instead.

Due to HTTP request size limitations, there is a limit of 100 mobile numbers for each message submitted.

`schedule_two_way_sms()`

This method will save the *message* passed in associating it with the *messageid* specified. The message is sent out at the date/time specified. The *messageid* can be used in later calls to ***get_reply_sms()*** to retrieve replies to this message and this message only. On success, this method returns the 32 byte ID used by the directSMS gateway to uniquely identify the submitted SMS message. If the operation fails, null is returned instead.

Due to HTTP request size limitations, there is a limit of 100 mobile numbers for each message submitted.

`cancel_scheduled_sms()`

This method will cancel the sending of the scheduled message identified by the ID specified. If the operation fails, null is returned instead.

Please note, only the user account that created the original scheduled message is allowed to cancel it.

`get_reply_sms()`

The method will return an array of ***reply_sms*** objects that represent the unread replies to 2-way SMS messages.

This method accepts 2 optional parameters:

messageid: This instructs the SMS gateway to retrieve all unread replies to the 2-Way SMS message identified by *messageid* (the parameter passed with the SMS when it was submitted using ***send_two_way_sms()***). If this parameter is not present, unread replies to **ALL** 2-way messages are returned.

mark_as_read: This parameter instructs the gateway to mark all reply messages returned as **read**. This stops the gateway from returning them again in the future. If this parameter is not present or is set to `false` the returned messages are left as is and are in fact returned again in future calls to this method.

`get_inbound_sms()`

The method will return an array of ***inboun_sms*** objects that represent the unread inbound SMS messages.

This method accepts 2 optional parameters:

inbound_number: This instructs the SMS gateway to retrieve all unread inbound SMS messages received on the specified number. If the number is not specified, the unread messages sent to ALL inbound numbers leased by the customer are retrieved.

mark_as_read: This parameter instructs the gateway to mark all messages returned as **read**. This stops the gateway from returning them again in the future. If this parameter is not present or is set to `false` the returned messages are left as is and are in fact returned again in future calls to this method.

`disconnect()`

This method will disconnect the client from the gateway and close the connection.

reply_sms

This is the class of object returned by the directSMS API server in response to a request for **unread** 2-way message replies.

The attributes of this class are listed below:

message: This is the SMS message text sent as a reply to the original 2-way message.

messageid: The identifier that uniquely identifies this 2-Way message (in your system) that this reply belongs to. This is sent back to allow clients group replies to the different messages together. This is the *messageid* specified in your call to the ***get_reply_sms ()*** method to submit the 2-way message.

mobile: The mobile phone number of the person responding to this message. The number is returned in international format with a + at the front e.g. "+61412345678".

when: The number of seconds since this reply message was received by the gateway.

The following public methods are detailed below:

`to_string()`

This method returns a string representation of this reply object suitable for debugging.

inbound_sms

This is the class of object returned by the directSMS API server in response to a request for **unread** inbound messages.

The attributes of this class are listed below:

message: This is the SMS message text. The number is returned in international format with a + at the front e.g. “+61412345678”.

inbound_number: The number the SMS message was sent to.

mobile: The mobile phone number of the message sender. The number is returned in international format with a + at the front e.g. “+61412345678”.

when: The number of seconds since this message was received by the gateway.

The following public methods are detailed below:

`to_string()`

This method returns a `string` representation of this message object suitable for debugging.