



m-Qube SMS Gateway

**API Reference
v2.0**

SMSGATEWAYAPI v2.0

Last modified: February 17th, 2006

Copyright © 2006 m-Qube, Incorporated. All rights reserved. Printed in the USA.

This manual contains material that is protected by copyright laws. No part of this manual may be reproduced, transcribed, stored in a retrieval system, or translated into any language, or transmitted in any form whatsoever, without prior written consent of m-Qube. This manual may only be used by authorized users of m-Qube's application services, and then solely for the purpose of gaining a better understanding of the features of the m-Qube application services.

m-Qube has taken due care in preparing this manual. However, m-Qube does not warrant that this manual is error free. Nothing contained herein modifies or alters the standard terms and conditions of the applicable service or use agreement by which the application services were acquired, nor increases in any way m-Qube's liability. In no event will m-Qube or its affiliates or licensors be liable for incidental or consequential damages arising in connection with the use of the m-Qube application services, this manual, or any related materials, even if advised of the possibility of such damages.

The information in this manual is subject to change without notice.

m-Qube, the m-Qube logo, and m-Qube Studio are trademarks or registered trademarks of m-Qube Incorporated. All other company and product names may be trade names or trademarks of their respective owners.



Contents

1	Introduction	1
	Available Gateway Interfaces	1
2	SMS Gateway SMPP Interface	3
	SMPP Bind Configuration	3
	Connection Settings	3
	Message Rate Settings	4
	Resubmittable Errors	5
	Sending Standard MT Messages	5
	Sending Premium MT Messages	7
	Receiving Billing Receipts	8
	Requesting Billing Receipts	8
	Receiving Billing Receipts via SMPP	8
	Receiving Billing Receipts via HTTP	11
	Sending WAP Push Requests	12
	Constructing the WAP Push	12
	Sample WAP Push Message	14
	Executing the WAP Push	14
	Sending Parameters to m-Qube Studio	14
	Receiving MO Messages	15
	Acknowledging MO Messages	16
3	SMS Gateway Web Service Interface	18
	Sending Standard MT Messages	18
	Calling sendMessage via SOAP	19
	Calling sendMessage via HTTP	21
	sendMessage Error Codes	21
	Sending Premium MT Messages	22
	Receiving Billing Receipts	24
	Requesting a Billing Receipt	24
	Receiving a Billing Receipt	24
	Sending WAP Push Requests	25
	Sending Parameters to m-Qube Studio	26
	Checking a Mobile Number	27
	Calling checkMobileNumber via SOAP	27
	Calling checkMobileNumber via HTTP	28
	Bulk Sending MT Messages	29
	Bulk Send Job Lifecycle	30
	Bulk Send Client Application	31
	Job Submit Response	35
	Job Status Report	36
	Job Receipt Report	37
	Job Notifications	39
	Manually Submitting a Job Request	39
	Avoiding Duplicate Job Submissions	41
	Checking Field Size Limits	41
	Job Submission Status Codes	41
	Receiving MO Messages	43

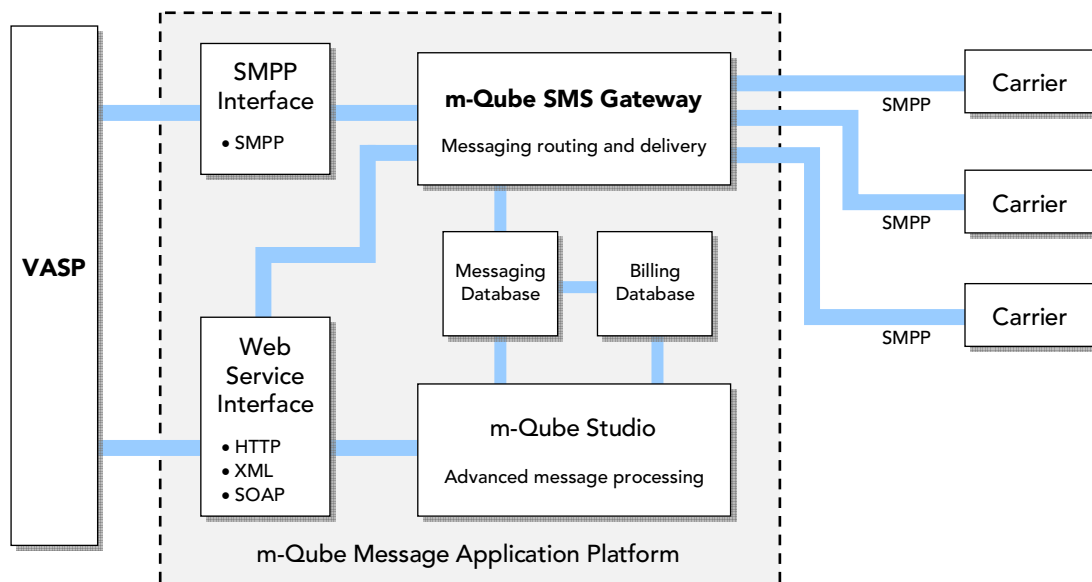


Appendix A: Carrier Information	45
Carrier IDs and Direct Connection Capabilities	45
Allowable Text Message Lengths	46
Appendix B: SMS Gateway Error Codes	48
Appendix C: Carrier-Specific Notes	56
Nextel Message Length	56
Sprint WAP Push URLs	56
Appendix D: Technical Support and Sales	58
m-Qube Technical Support	58
m-Qube Sales and Professional Services	58



1 Introduction

m-Qube SMS Gateway is the core message routing and delivery component of m-Qube's Message Application Platform. It gives content providers and value added service providers (VASPs) access to every major North American wireless carrier via m-Qube's own carrier-grade data center, automatically performing message routing to the appropriate destination carrier based on the subscriber phone number. Along with support for the latest message routing standards, SMS Gateway offers standard and premium billing services, billing notifications, receipt notifications, binary content delivery, and WAP push capabilities.



SMS Gateway also underpins m-Qube Studio, the other main component of the Message Application Platform. Studio is a web-based visual interface through which you can create, test, and deploy sophisticated interactive mobile messaging programs such as sweepstakes, polls, and trivia competitions.

NOTE

This manual covers m-Qube SMS Gateway only. For complete information about the capabilities of m-Qube Studio, please refer to the separate *m-Qube Studio User Guide* and the *m-Qube Studio API Reference*.

Available Gateway Interfaces

The m-Qube SMS Gateway supports two different messaging interfaces: SMPP and Web Service. Since not all of the Gateway functionality is available on both interfaces, you should analyze your messaging needs carefully before committing to one interface or the other. The following table shows the feature matrix for the SMPP and Web Service interfaces. It includes m-Qube Studio for reference, since some of the Gateway functionality is also achievable via the Studio API.



	SMS Gateway SMPP	SMS Gateway Web Service	Studio Web Service
Mobile Terminated (MT) Messages	Yes	Yes	Yes
Mobile Originated (MO) Messages	Yes	Yes	Yes
Premium Messages	Yes	Yes	Yes
Bulk Send	No	Yes	Yes
Delivery Receipts	No	Yes (Bulk Send only)	Yes
Billing Receipts	Yes	Yes	Yes
WAP Push	Yes	Yes	Yes
E-mail Routing	No	Yes	Yes
IM Routing	No	No	Yes
Carrier Number Lookup	No	Yes	Yes
Carrier ID in MO Messages	Yes	Yes	Yes
Studio Program Triggering	Yes	Yes	Yes



2 SMS Gateway SMPP Interface

The m-Qube SMS Gateway SMPP interface allows messaging applications to connect to SMS Gateway via a direct bind. The SMPP interface supports MT messages at both standard and premium rates. Billing receipts are optionally available for premium MT messages, while MO messages include carrier identification. You can also use the SMPP interface for WAP pushes and to pass parameters into m-Qube Studio. However, SMPP cannot be used to route messages to e-mail addresses or IM networks.

NOTE

This section of the manual assumes that you're already familiar with the SMPP protocol. For more information on SMPP, including the full SMPP 3.4 specification and links to client libraries and tools, please see the SMS Forum website at <http://www.smsforum.com>.

SMPP Bind Configuration

To use the m-Qube SMS Gateway via SMPP, you need to connect as a standard External Short Message Entity (ESME) as described in the SMPP v3.4 Specification. The preferred connection method is Transceiver, but you can also bind as a Transmitter/Receiver if necessary.

The following standard PDUs are employed for connecting, transmitting messages, and receiving messages:

Client to SMS Gateway	SMS Gateway to Client
bind_transceiver	bind_transceiver_resp
bind_transmitter	bind_transmitter_resp
bind_receiver	bind_receiver_resp
enquire_link	enquire_link
enquire_link_resp	enquire_link_resp
submit_sm	submit_sm_resp
deliver_sm_resp	deliver_sm
query_sm	query_sm_resp
unbind	unbind
unbind_resp	unbind_resp

The relevant settings are detailed below. If you need SMS Gateway to accept additional PDU types or if you require access to specific parameters, please contact the m-Qube Connectivity Team.

Connection Settings

Set the body parameters of the *bind_transceiver* PDU as follows:



bind_transceiver	
Parameter	Value
system_id	As supplied by m-Qube
password	As supplied by m-Qube
system_type	0x00
interface_version	0x34 (v3.4)
addr_ton	0x00
addr_npi	0x00
address_range	0x00

When disconnecting, always send an *unbind* PDU and wait for an unbind response before closing the socket connection. Unless you've specifically arranged multiple connections, only one SMPP bind is allowed at a time. Therefore if you don't send an *unbind* PDU, you may find that attempts to immediately rebind are denied.

Your bind should be up and ready to accept messages at all times and you have responsibility for rebinding if the connection is dropped. Note that if the connection is not available, m-Qube will only queue up to a certain number of MO messages (roughly 100,000). Beyond that limit, messages are discarded.

To ensure a constant connection, send an *enquire_link* PDU every 30 seconds. If you don't receive an immediate response, you should assume the connection has dropped and attempt to rebind.

Message Rate Settings

If configurable on your platform, set the message window size and MT message rate as follows:

Setting	Value	Definition
Window Size	10 msgs	The maximum number of outstanding (i.e., unacknowledged) SMPP operations that are allowed at the same time.
MT Message Rate	10 msgs/sec	The maximum rate at which SMS Gateway is set to receive messages via SMPP. m-Qube can raise this limit by request.

If you exceed the MT message rate, SMS Gateway may reject your messages and reply with the following throttle rate error in the *command_status* header field of the *submit_sm_resp* PDU:

0x58 - ESME_RTHROTTLED

In this case, you should pause the message stream for one second for each error response received.

By default, the rate of MO traffic sent from SMS Gateway is unrestricted. If you need temporary MO throttling, send back the ESME_RTHROTTLED error in the *deliver_sm_resp* PDU. SMS Gateway will pause one second for each error sent. If you want to adjust the MO message rate permanently, please notify m-Qube.



Resubmittable Errors

In the event of an SMPP error, SMS Gateway returns a code in the `command_status` header field of the response message, as described in the SMPP 3.4 specification. The errors for which you should resubmit a message are listed in the table below, along with the suggested delay (in milliseconds), the retry limit, and the incremental delay (in milliseconds).

Event Code	Description	Delay (ms)	Limit	Inc-delay (ms)
0x0004	ESME_RINVBNDSTS	30000	10	0
0x0008	ESME_RSYSERR	10000	100	1000
0x000D	ESME_RBINDFAIL	30000	10	0
0x0011	ESME_RCANCELFAIL	30000	10	0
0x0013	ESME_RREPLACEFAIL	30000	10	0
0x0014	ESME_RMSGQFUL	10000	None	1000
0x0045	ESME_RSUBMITFAIL	10000	10	1000
0x0058	ESME_RTHROTTLED	10000	None	5000
0x0067	ESME_RQUERYFAIL	30000	10	0
0x00FE	ESME_RDELIVERYFAILURE	10000	100	1000

Any other error code means the request was invalid and you should not attempt to resubmit. If required, m-Qube can customize the retry policy for specific errors.

Sending Standard MT Messages

Send your MT messages via a `submit_sm` PDU configured as shown in the table below. Note that all other non-header parameters in the PDU must be set to NULL.

submit_sm	
Parameter	Value
source_addr	The service code you want to use for this message.
destination_addr	The destination number for this message, formatted as a 10-digit North American Numbering Plan (NANP) number. For example: 5008885555 The number must not include any '+' (or '+1') characters, other alpha characters, symbols, or spaces. If the destination number is not a valid mobile subscriber number, SMS Gateway will reject the message.
esm_class	Use only if the message is a WAP Push. See Sending WAP Push Requests for details. Otherwise, set to 0x00.



registered_delivery	Use only if you require a delivery receipt. See Receiving Billing Receipts for details. Otherwise, set to 0x00.
data_coding	Set only if the message is a WAP Push. See Sending WAP Push Requests for details. Otherwise, set to 0x00.
sm_length	Length of the short_message data in octets.
short_message	The text of the message. The SMPP protocol supports up to 254 octets of data. However, the practical maximum message length varies by carrier. See Appendix A: Carrier Information for allowable message lengths.
product code (0x1520)	An optional m-Qube-specific parameter used if this is a premium message. See the Sending Premium MT Messages section for more information.
client message ID (0x1550)	An optional m-Qube-specific parameter used to override the default message ID normally supplied by SMS Gateway.

SMS Gateway will respond with a standard *submit_sm_resp* PDU as described in the SMPP 3.4 specification.

submit_sm_resp	
Parameter	Value
command_status	For a successful submit, the command_status field is set to 0. For an unsuccessful submit, the field contains the relevant SMPP error. (See Resubmittable Errors above.)
message_id	For a successful submit, the message_id field contains a unique ID (either assigned automatically by SMS Gateway or else matching the optional client message ID passed via the submit_sm). For an unsuccessful submit, the field is null.

NOTE

SMS Gateway uses the message ID returned in the *submit_sm_resp* PDU to reference the original message in subsequent operations such as billing receipts. If you plan to reconcile the message against the receipt, your application will need to make note of the message ID transmitted in the *submit_sm_resp* PDU.

Sample submit_sm

```
00 00 00 3B 00 00 00 04 00 00 00 00 00 00 01 .....
00 00 00 31 32 33 34 35 00 00 00 35 30 30 38 38 ...12345...50088
38 35 35 35 35 00 00 00 00 00 00 00 00 00 0B 85555.....
48 65 6C 6C 6F 20 57 6F 72 6C 64                Hello world
```

The relevant parts of the message decode as follows:

```
command_length: 59
command_id: 4 (submit_sm)
source_addr: 12345
```



```
destination_addr: 5008885555
sm_length: 11
short_message: Hello world
```

Sample submit_sm_resp

```
00 00 00 2d 80 00 00 04 00 00 00 00 00 00 08 .....
30 6a 36 74 39 75 30 31 6f 64 69 69 76 61 30 37 0j6t9u0lodiiva07
6d 63 34 34 68 33 30 69 39 66 33 67 00          mc44h30i9f3g.
```

The relevant parts of the message decode as follows:

```
command_length: 45
command_id: 80000004 (submit_sm_resp)
command_status: 0 (no error)
message_id: 0j6t9u0lodiiva07mc44h30i9f3g
```

Sending Premium MT Messages

To send premium messages via the SMS Gateway SMPP interface, simply flag the message as premium billable then use the same basic *submit_sm* PDU configuration described in the [Sending Standard MT Messages](#) section above. There are two ways to flag premium messages: [Product Codes](#) and [Short Code Suffixes](#).

Product Codes

The preferred method to flag a message as premium-billable using the SMPP interface is to set the optional m-Qube-specific product code parameter in the *submit_sm* PDU.

submit_sm	
Parameter	Value
product code (0x1520)	A text string that defines the pricing level for the premium message. Product codes are set up by m-Qube to meet your specific needs. In addition to setting the price, they typically also indicate the content provider, short code, item name, and purchase type (subscription or individual). For example: ACMEX_12345_RINGHIP_999_S ACMEX_12345_RINGHIP_199_I ACMEX_12345_WALLPAP_999_S

Short Code Suffixes

As an alternative to sending a product code in the *submit_sm*, you can also arrange with m-Qube to use a short code suffix to trigger premium billing.

For example, SMS Gateway can be configured so that messages sent to short code 12345 are standard rated, messages sent to 123451 are provisioned as premium rated at the first price point, messages sent to 123452 are provisioned at the second premium price point, and so on. These premium short code suffixes are only used internally between you and m-Qube; carrier traffic will either appear as 12345, or as a carrier-specific short code.



Receiving Billing Receipts

For premium MT messages, the SMS Gateway SMPP interface allows you to request a billing receipt indicating the outcome of the billing transaction between SMS Gateway and the carrier. Billing receipts are normally returned via SMPP but you can optionally request m-Qube to provide receipts via an HTTP post mechanism.

This billing receipts feature is based upon the standard SMPP delivery receipt functionality. Please refer to the SMPP Specification v3.4 for a complete description of delivery receipt behavior on 3.4 connections.

NOTE

Future releases of SMS Gateway will also support ordinary delivery receipts for standard-rated messages.

Requesting Billing Receipts

To request billing receipts from SMS Gateway, set the `registered_delivery` parameter in the `submit_sm` PDU as indicated in the table below.

submit_sm	
Parameter	Value
registered_delivery	0x20

The message must also be identified as premium rated. (See [Sending Premium MT Messages](#).)

Receiving Billing Receipts via SMPP

Billing receipts are delivered from SMS Gateway via a standard `deliver_sm` PDU with the `esm_class` parameter set to 0x04 to indicate that `short_message` contains an SMSC delivery receipt. (See Appendix B of the SMPP 3.4 specification.)

General status information is returned via the `message_state` and `receipted_message_id` optional parameters in the `deliver_sm` PDU.

deliver_sm	
Parameter	Value
message_state (0x0427)	Indicates the overall status of the message. Set to 2 (DELIVERED) for billing success. Set to 3 (EXPIRED) or 5 (UNDELIVERABLE) in the case of a billing or delivery error.
receipted_message_id (0x001E)	Corresponds to the ID of the <code>deliver_sm_resp</code> PDU you received from SMS Gateway when you sent the original message. Used to reconcile the billing receipt with the message to which it applies.

In addition, the following six vendor-specific parameters are set in the `deliver_sm` PDU to return more information about the billing event.



deliver_sm	
Parameter	Value
output_code (0x1530)	A three-character string indicating the result of the billing event. Possible values are as follows: NRT - No route, could not deliver message DLV - Delivery error OPT - Opted out PNC - Subscriber's carrier not supported for premium billing PBS - Premium billing succeeded PBD - Premium billing declined PBI - System error during processing PBO - Premium billing declined, subscriber is over billing limit EXP - Message expired, no delivery confirmation within 72 hours BLO - Blocked If billing failed, the error_message parameter contains a more detailed description of the problem.
error_message (0x1531)	A description of the billing error.
auth_code (0x1532)	The premium authorization return code. A value of 0 indicates the premium authorization was successful. For failed authorizations, the value is dependent on the billing method. For example, on Qpass-supported carriers auth_code is set to the error code returned by Qpass. See Appendix B: SMS Gateway Error Codes for a complete list of errors.
auth_message (0x1533)	The error message for a failed premium authorization. For example: No Phone Number: Subscriber phone number was not matched in the Qpass system or AWS Portal If authorization was successful, auth_message is blank.
carrier_id (0x1402)	The carrier ID. See Appendix A: Carrier Information for a complete list of carrier IDs.
carrier_name (0x1404)	The full name of the carrier.

The delivery receipt contained in the short_message parameter includes the eight standard fields defined in the SMPP specification, plus three additional m-Qube fields (cid, cerr, and src) that duplicate some of the information returned in the PDU parameters listed above. For example:

```
id:0j6t9u000k01tv073e4413t044h0 sub:001 dlvrd:001 submit date:0512121539
done date:0512121539 stat:DELIVRD err:PBS text:Hello world cid:5 cerr:PBS src:GWY
```

Field	Length	Field Type	Value
id	28	Standard	Same as the receipted_message_id (0x001E) parameter.
sub	3	Standard	Always set to 001 for billing receipts.



dlvrd	3	Standard	Always set to 001 for billing receipts.
submit date	10	Standard	The message submit date in YYMMDDmmhh format.
done date	10	Standard	The message completion date in YYMMDDmmhh format.
stat	7	Standard	Same as the message_state (0x0427) parameter.
err	3	Standard	Same as the output_code (0x1530) parameter.
text	12	Standard	The first 12 characters of the original message.
cid	3	m-Qube	Same as the carrier_id (0x1402) parameter.
cerr	6	m-Qube	Same as the auth_code (0x1532) parameter.
src	3	m-Qube	A three-character code indicating the source of the billing event: GWY - Gateway billing SUB - Subscription billing STU - Studio billing

The complete *deliver_sm* for a billing receipt looks like this:

```
00 00 01 33 00 00 00 05 00 00 00 00 00 00 0c .....
00 00 00 35 30 30 38 38 38 35 35 35 35 00 00 00 ...5008885555...
32 30 30 35 00 04 00 00 00 00 00 00 00 00 9b 69 2005.....i
64 3a 30 6a 36 74 39 75 30 30 30 6b 30 31 74 76 d:0j6t9u000k01tv
30 37 33 65 34 34 31 33 74 30 34 34 68 30 20 73 073e4413t044h0 s
75 62 3a 30 30 31 20 64 6c 76 72 64 3a 30 30 31 ub:001 dlvrd:001
20 73 75 62 6d 69 74 20 64 61 74 65 3a 30 35 31 submit date:051
32 31 32 31 35 33 39 20 64 6f 6e 65 20 64 61 74 2121539 done dat
65 3a 30 35 31 32 31 32 31 35 33 39 20 73 74 61 e:0512121539 sta
74 3a 44 45 4c 49 56 52 44 20 65 72 72 3a 50 42 t:DELIVRD err:PB
53 20 74 65 78 74 3a 48 65 6c 6c 6f 20 57 6f 72 s text:Hello wor
6c 64 20 20 63 69 64 3a 35 20 63 65 72 72 3a 50 1d cid:5 cerr:P
42 53 20 73 72 63 3a 47 57 59 04 27 00 01 02 00 BS src:GWY.....
1e 00 1d 30 6a 36 74 39 75 30 30 30 6b 30 31 74 ...0j6t9u000k01t
76 30 37 33 65 34 34 31 33 74 30 34 34 68 30 00 v073e4413t044h0.
14 02 00 01 35 14 04 00 06 53 70 72 69 6e 74 15 ....5....Sprint.
30 00 03 50 42 53 15 31 00 00 15 32 00 01 30 15 0..PBS.....0.
33 00 00 15 50 00 1c 30 6a 36 74 39 75 30 30 6a .....0j6t9u00j
37 39 65 75 6e 30 37 33 65 34 34 31 33 74 30 34 79eun073e4413t04
38 71 67                                     8qq
```

The relevant parts of the message decode as follows:

Command length: 307
Command ID: 5 (deliver_sm)
Source address: 5008885555
Destination address: 2005
ESM class: 4
Optional TLV 0x427: 2
Optional TLV 0x1e: 0j6t9u000k01tv073e4413t044h0
Optional TLV 0x1402: 5



```
Optional TLV 0x1404: Sprint
Optional TLV 0x1530: PBS
Optional TLV 0x1531: null
Optional TLV 0x1532: 0
Optional TLV 0x1533: null
Optional TLV 0x1550: 0j6t9u00j79eun073e4413t048qg
Message:
id:0j6t9u000k01tv073e4413t044h0 sub:001 dlvr:001 submit date:0512121539
done date:0512121539 stat:DELIVRD err:PBS text:Hello world cid:5 cerr:PBS src:GWY
```

Receiving Billing Receipts via HTTP

As an alternative to sending billing receipts through SMPP, m-Qube can provision SMS Gateway to deliver them via an HTTP POST to a provided URL. The following fields are included in an HTTP billing receipt:

Parameter	Description
destination	Phone number of the subscriber in UMDA format. For example: tel:5008885555
serviceCode	The service code used for the message.
authCode	The premium authorization return code. A value of 0 indicates the premium authorization was successful. For failed authorizations, the value is dependent on the billing method. For example, on Qpass-supported carriers authCode is set to the error code returned by Qpass. See Appendix B: SMS Gateway Error Codes for a complete list of errors.
authMessage	The error message for a failed premium authorization. For example: No Phone Number: Subscriber phone number was not matched in the Qpass system or AWS Portal If authorization was successful, authMessage is blank.
carrierName	The full name of the carrier.
messageID	An internal m-Qube ID assigned the message. (No longer used - see transaction ID below instead.)
outputCode	A three-character string indicating the result of the billing event. Possible values are as follows: NRT - No route, could not deliver message DLV - Delivery error OPT - Opted out PNC - Subscriber's carrier not supported for premium billing PBS - Premium billing succeeded PBD - Premium billing declined PBI - System error during processing PBO - Premium billing declined, subscriber is over billing limit EXP - Message expired, no delivery confirmation within 72 hours BLO - Blocked



transactionID	Corresponds to the ID of the <i>deliver_sm_resp</i> you received from SMS Gateway when you sent the original message. Used to reconcile the billing receipt with the message to which it applies.
---------------	---

The data is transmitted in the body of the POST as a string of key-value pairs. For example:

```
destination=tel:6175551212&serviceCode=3547&authCode=0&authMessage=&carrierName=NEXTEL
&messageID=1m7dn6h02039u00dt444b1r61db0&outputCode=PBS&transactionID=1m7dn6h02039u00dt
444b1r61db0
```

It's your responsibility to have a server program running at the specified URL that will accept the HTTP POST. If a receiving program isn't present, or if the POST fails for any reason, SMS Gateway will retry delivery three times over a period of 15 minutes. After three failed delivery attempts, the receipts are placed in a dead letter queue. Upon request, m-Qube will attempt to resend the queued receipts later.

Sending WAP Push Requests

If your connection with m-Qube is provisioned to support it, you can send WAP Push requests via SMPP across participating carriers. SMS Gateway will recognize the request and execute the appropriate carrier-specific WAP Push on your behalf. Some carriers also have additional restrictions on WAP pushes. See [Appendix C: Carrier-Specific Notes](#) for more information.

NOTE

SMS Gateway supports WAP Push via SMPP primarily to accommodate SMPP-centric organizations. If your messaging applications do not need to be exclusively SMPP-based, the Web Service interface provides a simpler WAP Push mechanism. Please see Chapter 3: [SMS Gateway Web Service Interface](#) for details.

This section provides an overview of how to send a WAP Push via SMS Gateway using SMPP. For more detailed background information, please consult the following WAP Specification documents:

- WAP Service Indication Specification (WAP-167-ServiceInd-20010731-a)
- Wireless Session Protocol Specification (WAP-230-WSP-20010705-a)
- Wireless Datagram Protocol Specification (WAP-259-WDP-20010614-a)

These documents are available in PDF format from the Open Mobile Alliance website:

<http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html>

Constructing the WAP Push

An SMS Gateway WAP Push is achieved by placing a binary encoded Service Indication (SI) into the `short_message` parameter of a *submit_sm* PDU. The message is comprised of four main pieces.

[submit_sm Parameters](#)
[User Data Header](#)
[WTP Header](#)
[Service Indication](#)

NOTE

The current version of SMS Gateway requires that WAP Push requests are encoded



in a single SMPP message. This limits the combined size of the title and the WAP URL.

submit_sm Parameters

Set the *submit_sm* PDU parameters normally as described in the [Sending Standard MT Messages](#) section, with the following two additions:

- Set the *esm_class* parameter to 0x40. This turns on the User Data Header Indicator (UDHI) flag, indicating that the beginning of the *short_message* data is a header. (See section 5.2.12 of the SMPP v3.4 Specification.)
- Set the *data_coding* parameter to 0xF4. This changes the SMPP data coding scheme of the *short_message* contents to GSM message class control, 8-bit binary. (See section 5.2.19 of the SMPP v3.4 Specification.)

For example:

```
00 00 00 60 00 00 00 04 00 00 00 00 00 00 00 01
00 00 00 31 32 33 34 35 00 00 00 35 30 30 38 38
38 35 35 35 35 00 40 00 00 00 00 00 00 00 f4 00
```

User Data Header

The *short_message* parameter must begin with a User Data Header (UDH) string that defines the destination port as 2948 and the source port as 9200:

```
06 05 04 0B 84 23 F0
```

The seven octets translate as:

06 = UDH length
05 = Identifier for application port addresses
04 = Header length
0B 84 = Destination port 2948
23 F0 = Source port 9200

See WAP-259-WDP-20010614-a for additional information about how User Data Headers are used in WAP.

WTP Header

The UDH is followed by a WTP header string. For example:

```
02 06 01 ae
```

The four octets translate as:

02 = Transaction ID (Push ID)
06 = PDU type (Push PDU)
01 = Header length
AE = Well-known header value token for "Content-type: application/vnd.wap.sic"

See WAP-230-WSP-20010705-a, section 8.2 "Protocol Data Unit Structure" for more information.



Service Indication

The WTP header is followed by a Service Indication containing the URL and message text, tokenized using WBXML encoding. For example, this Service Indication XML yields the WBXML string shown below.

```
<?xml version="1.0"?>
<!DOCTYPE si PUBLIC "-//WAPFORUM//DTD SI 1.0//EN"
"http://www.wapforum.org/DTD/si.dtd">
<si>
<indication href="http://wap.yahoo.com">m-Qube Msg</indication>
</si>
```

```
03 05 6A 00 45 C6 0C 03 77 61 70 2E 79 61 68 6F 6F 2E 63 6F 6D 00 01 03 6D 2D 51 75 62
65 20 4D 73 67 00 01 01
```

See WAP-167-ServiceInd-20010731-a for more information about Service Indications and WBXML encoding.

Sample WAP Push Message

This example shows the *submit_sm* parameters, User Data Header, WTP header, and Service Indication binary combined to form the complete SMS.

```
00 00 00 60 00 00 00 04 00 00 00 00 00 00 01 .....
00 00 00 31 32 33 34 35 00 00 00 35 30 30 38 38 ...12345...50088
38 35 35 35 35 00 40 00 00 00 00 00 00 f4 00 30 85555.....
06 05 04 0b 84 23 f0 02 06 01 ae 03 05 6a 00 45 .....
c6 0c 03 77 61 70 2e 79 61 68 6f 6f 2e 63 6f 6d ...wap.yahoo.com
00 01 03 6d 2d 51 75 62 65 20 4d 73 67 00 01 01 ...m-Qube Msg...
```

Executing the WAP Push

In order for SMS Gateway to distinguish a WAP push from a regular SMS message, the *source_addr* parameter in the *submit_sm* PDU must be a service code that m-Qube has specially provisioned to handle WAP push requests. Typically this takes the form of a one- or two-digit extension to an existing service code. Please contact the m-Qube Connectivity Team for more information.

Sending Parameters to m-Qube Studio

If desired, you can pass parameters to trigger an m-Qube Studio program via the SMS Gateway SMPP interface. Studio programs can add a wide range of customization to your MT messages. Please refer to the *m-Qube Studio User Guide* for a full description of the available Studio functionality.

To send parameters to Studio via SMPP, set the *source_addr* parameter in the *submit_sm* to be the service code of the Studio program you want to trigger. Then add the following m-Qube-specific optional parameter to the end of the PDU:



submit_sm	
Parameter	Value
smx_properties (0x151A)	The parameters expected by the Studio program, formatted as key-value pairs delimited by the "=" character and separated with an \n and/or an \r character between pairs. For example: firstname=John\nlastname=Smith

Receiving MO Messages

For the SMPP interface, SMS Gateway delivers MO messages via a standard *deliver_sm* PDU configured as follows:

deliver_sm	
Parameter	Value
source_addr	The mobile number from which the message originated.
destination_addr	The service code used for the message.
sm_length	Length of the short_message data in octets.
short_message	The text of the message.

To enable you to identify the carrier, the *deliver_sm* also includes two m-Qube-specific optional parameters: *carrier_id* and *carrier_name*.

deliver_sm	
Parameter	Value
carrier_id (0x1402)	String representing the numerical value of the carrier ID. See Appendix A: Carrier Information for the full list of carrier IDs.
carrier_name (0x1404)	The name of the carrier.

The following example shows an MO delivered from carrier 2 (T-Mobile):

```
00 00 00 4B 00 00 00 05 00 00 00 00 00 00 00 01 .....
00 00 00 35 30 30 38 38 38 35 35 35 35 00 00 00 ...5008885555...
31 32 33 34 35 00 00 00 00 00 00 00 00 00 00 0B 12345.....
48 65 6C 6C 6F 20 57 6F 72 6C 64 14 02 00 01 32 Hello world....2
14 04 00 07 54 4d 6f 62 69 6c 65 .....TMobile
```

The relevant parts of the message decode as follows:

```
Command length:75
Command ID: 5 (deliver_sm)
Source address: 5008885555
```



Destination address: 12345
Message: Hello world
Optional TLV 0x1402: 2
Optional TLV 0x1404: TMobile

Acknowledging MO Messages

You must reply with a *deliver_sm_resp* PDU for all MO messages sent to you. If a message is unacceptable to your system, please reply with an appropriate SMPP error code. If you do not reply with a *deliver_sm_resp*, SMS Gateway will continue to attempt delivery only until the queue reaches a certain size. (See [Connection Settings](#).)





3 SMS Gateway Web Service Interface

The m-Qube SMS Gateway Web Service Interface supports both individual and bulk send transmission of MT messages at either standard or premium rates. Delivery receipts and premium billing notifications are optionally available for MT messages, and individual messages can be routed to e-mail addresses as well as mobile numbers. MO messages are handled via an HTTP POST. You can also use the Web Service interface for WAP pushes and to pass parameters into m-Qube Studio.

The SMS Gateway Web Service interface is comprised of five separate methods, each of which is described in detail in this chapter.

Web Service Method	Description	Call Methods
sendMessage	Sends an individual MT message to a mobile number or e-mail address. See Sending Standard MT Messages and Sending Premium MT Messages .	HTTP POST/GET SOAP
WSGWXML	Receives and processes bulk send job requests sent either manually or by the Bulk Send Client application. See Bulk Sending MT Messages .	HTTP POST
WSGWStatus	Checks to see if a bulk send job is already in the queue. See Avoiding Duplicate Job Submissions .	HTTP GET
WSGWConfig	Shows the current field size limits for a bulk send job request. See Checking Field Size Limits .	HTTP GET
checkMobileNumber	Returns the carrier ID associated with a specified mobile number. See Checking a Mobile Number .	HTTP POST/GET SOAP

Sending Standard MT Messages

The `sendMessage()` Web service method allows you to send individual MT messages to a mobile number or e-mail address. `sendMessage()` is callable through SOAP (Java and .NET versions) or via an HTTP POST or GET operation. Each method is described in detail below.

For both the SOAP and HTTP implementations, `sendMessage()` takes the following input parameters.

Parameter	Description
interpret	Tells SMS Gateway to process any extra parameters passed in with the call. (See <code>extraParams</code> below). Always set <code>interpret</code> to true unless otherwise instructed by m-Qube.



destination	The destination phone number or e-mail address in Universal Mobile Device Address (UMDA) format. For example: tel:5008885555 or mailto:jdoe@example.com For HTTP calls, the destination value must be URL-encoded. For example: tel%3A5008885555 mailto%3Ajdoe@example.com
serviceCode	The service code for this message.
message	The text message. For HTTP calls, the message value must be URL-encoded. For example: Hello+Wireless+World
extraParams	Extra parameters passed in as part of the sendMessage() call. The dialog.op extra parameter is required and should always be set to sms_push for ordinary MT messages. Other optional extra parameters are used in conjunction with dialog.op to define special message processing. See the Sending Premium MT Messages, Sending WAP Push Requests, and Sending Parameters to m-Qube Studio sections for details. For Java SOAP clients, pass extraParams as a SOAP Map of <item> elements, each containing a key/value element pair. For example: <pre><extraParams> <item> <key>param1</key> <value>value1</value> </item> <item> <key>param2</key> <value>value2</value> </item> </extraParams></pre> For .NET SOAP clients and HTTP POST/GET calls, pass the extra parameters as a URL-encoded string. For example: extraParam1=extraValue1&extraParam2=extraValue2

If a sendMessage() call is successful, it returns a unique 28-character message GUID (Globally Unique Identifier), confirming that SMS Gateway has accepted the MT message for processing. For message ID examples, see the [sendMessage HTTP Response](#) and [sendMessage SOAP Response](#) sections.

Calling sendMessage via SOAP

To call sendMessage() via SOAP, construct your SOAP XML with the input parameters contained inside a <sendMessage> operation tag:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <sendMessage>
```



```
<interpret>true</interpret>
<destination>tel:5008885555</destination>
<serviceCode>12345</serviceCode>
<message>Hello world</message>
<extraParams>
  <item>
    <key>dialog.op</key>
    <value>sms_push</value>
  </item>
</extraParams>
</sendMessage>
</soapenv:Body>
</soapenv:Envelope>
```

Post the XML to the following Web service endpoint:

<https://studio.mqube.us/campaignapp/services/MessagingWS>

Note that you also need to pass your m-Qube user ID and password for basic authentication.

If you want to examine the `sendMessage()` interface or generate stub files for your applications, a corresponding WSDL file is available here:

<https://studio.mqube.us/campaignapp/services/MessagingWS?WSDL>

.NET Version

A separate instance of `sendMessage()` for .NET clients is also available:

<https://studio.mqube.us/campaignapp/services/MessagingWS2>

<https://studio.mqube.us/campaignapp/services/MessagingWS2?WSDL>

In the .NET version, `extraParams` values are passed as a URL-encoded string rather than as a SOAP Map. For example:

```
<extraParams>dialog.op=sms_push</extraParams>
```

sendMessage SOAP Response

If the `sendMessage()` call is successful, the Web service returns a SOAP response containing a 28-character message ID:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <sendMessageResponse
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <sendMessageReturn
        xsi:type="xsd:string">1j0j9u011760bk00tm43v31cpef0
      </sendMessageReturn>
    </sendMessageResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

If the `sendMessage()` call fails, the Web service returns a SOAP Fault. For example:



```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>soapenv:Server.userException</faultcode>
      <faultstring>com.proteusmobile.campaignapp.StudioException: 03573- Invalid
        or non deliverable UMDA tel:6170000000</faultstring>
      <detail/>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

See [sendMessage Error Codes](#) for a list of possible errors.

Calling sendMessage via HTTP

To call `sendMessage()` via HTTP, send a POST request to the following URL:

`http://studio.mqube.us/campaignapp/post/sendMessage`

Pass the input parameters within the body of the HTTP request as a URL-encoded string. For example:

```
interpret=true&destination=tel%3A5008885555&serviceCode=12345&message=
Hello+world&dialog.op=sms_push
```

Note that you also need to pass your m-Qube user ID and password for basic authentication.

For interactively testing in a browser, you can also call `sendMessage()` with an HTTP GET by passing the input parameters appended to the URL. For example:

```
http://studio.mqube.us/campaignapp/post/sendMessage?interpret=true&destination=
tel%3A5008885555&serviceCode=12345&message=Hello+world&dialog.op=sms_push
```

sendMessage HTTP Response

If the `sendMessage()` call is successful, the body of the HTTP response contains a 28-character message ID. For example:

```
Message Id: 1j0j9u0002bres006s43i3iu9mi0
```

If the `sendMessage()` call fails, the HTTP response header contains the relevant error code. For example:

```
HTTP/1.1 500 03573- Invalid or non deliverable UMDA tel:6170000000
```

See [sendMessage Error Codes](#) for a list of possible errors.

sendMessage Error Codes

If the `sendMessage()` call fails, the Web service returns an error code and brief description. The following table lists the most common error conditions.



Error	Cause
00000 Running program not found	The serviceCode value is not a valid provisioned service code.
03573 Invalid or non-deliverable UMDA	The mobile phone number specified in the destination parameter is invalid or not routable.
11100 Not a recognized UMDA transport	The destination parameter contains an unrecognized or misspelled UMDA prefix. All destination addresses must begin with either tel: or mailto:.
11103 Invalid UMDA format	The destination parameter contains an incorrectly formatted mobile phone number or e-mail address.
11104 Cannot construct an UMDA from null arguments	The destination parameter is empty.
11120 Phone number is too long	Either the destination mobile number or the serviceCode value exceeds the maximum 10 digits.
11121 Invalid phone number characters	The serviceCode parameter contains invalid characters.
11122 Phone number is null	The serviceCode parameter is empty.
11123 Phone number has invalid format	The destination phone number contains invalid characters.
11124 Phone number is too short	The destination phone number is too short or the serviceCode parameter is missing or incomplete.
Unknown Host Exception	The destination parameter contains an invalid e-mail domain name.
Address must start with an alphanumeric character	The destination e-mail address starts with an invalid character.
HTTP 500 Error	The server is temporarily unavailable.

Sending Premium MT Messages

To send premium messages via the SMS Gateway Web Service interface, simply flag the message as premium billable then use the same `sendMessage()` Web service call described in the [Sending Standard MT Messages](#) section above. There are two ways to flag premium messages: [Product Codes](#) and [Short Code Suffixes](#).



Product Codes

The preferred method to identify a message as premium-billable using the Web Service interface is to pass the `dialog.op` and `dialog.psms_pricecode` extra parameters in a `sendMessage()` call as shown in the table below. (See [Sending Standard MT Messages](#) for the complete list of `sendMessage()` input parameters and instructions on how to format `extraParams` for SOAP and HTTP.)

Parameter	Value
extraParams	<code>dialog.op</code> Set to <code>psms_push</code> to indicate that this is a premium message. <code>dialog.psms_pricecode</code> Set to a text string that defines the pricing level for the premium message. Product codes are set up by m-Qube to meet your specific needs. In addition to setting the price, they typically also indicate the content provider, short code, item name, and purchase type (subscription or individual). For example: <code>ACMEX_12345_RINGHIP_999_S</code> <code>ACMEX_12345_RINGHIP_199_I</code> <code>ACMEX_12345_WALLPAP_999_S</code>

SOAP example:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <sendMessage>
      <interpret>true</interpret>
      <destination>tel:5008885555</destination>
      <serviceCode>12345</serviceCode>
      <message>This is a Premium Message</message>
      <extraParams>
        <item>
          <key>dialog.op</key>
          <value>psms_push</value>
        </item>
        <item>
          <key>dialog.psms_pricecode</key>
          <value>ACMEX_12345_RINGHIP_999_S</value>
        </item>
      </extraParams>
    </sendMessage>
  </soapenv:Body>
</soapenv:Envelope>
```

HTTP POST example:

```
interpret=true&destination=tel%3A5008885555&serviceCode=12345&message=This+is+a
+Premium+Message&dialog.op=psms_push&dialog.psms_pricecode=ACMEX_12345_RINGHIP_999_S
```



Short Code Suffixes

As an alternative to sending a product code in the `submit_sm`, you can also use a pre-arranged short code suffix to trigger premium billing.

For example, SMS Gateway can be configured so that messages sent to short code 12345 are standard rated, messages sent to 123451 are provisioned as premium rated at the first price point, messages sent to 123452 are provisioned at the second premium price point, and so on. These premium short code suffixes are only used internally between you and m-Qube; carrier traffic will either appear as 12345, or as a carrier-specific short code.

Receiving Billing Receipts

For premium MT messages, the SMS Gateway Web Service interface allows you to request a billing receipt indicating the outcome of the billing transaction between SMS Gateway and the carrier.

Requesting a Billing Receipt

To request billing receipts for messages submitted via the `sendMessage()` Web service call, you need to use a service code that has been provisioned specifically for billing receipts. Please contact the m-Qube Connectivity Team for more information.

The message must also be identified as premium rated. (See [Sending Premium MT Messages](#).)

Receiving a Billing Receipt

For the SMS Gateway Web Service interface, billing receipts are delivered via an HTTP POST to a provided URL. The following fields are included in an HTTP billing receipt:

Parameter	Description
destination	Phone number of the subscriber in UMDA format. For example: tel:5008885555
serviceCode	The service code used for the message.
authCode	The premium authorization return code. A value of 0 indicates the premium authorization was successful. For failed authorizations, the value is dependent on the billing method. For example, on Qpass-supported carriers authCode is set to the error code returned by Qpass. See Appendix B: SMS Gateway Error Codes for a complete list of errors.
authMessage	The error message for a failed premium authorization. For example: No Phone Number: Subscriber phone number was not matched in the Qpass system or AWS Portal If authorization was successful, authMessage is blank.
carrierName	The full name of the carrier.
messageID	An internal m-Qube ID assigned the message. (No longer used - see transaction ID below instead.)



outputCode	A three-character string indicating the result of the billing event. Possible values are as follows: NRT - No route, could not deliver message DLV - Delivery error OPT - Opted out PNC - Subscriber's carrier not supported for premium billing PBS - Premium billing succeeded PBD - Premium billing declined PBI - System error during processing PBO - Premium billing declined, subscriber is over billing limit EXP - Message expired, no delivery confirmation within 72 hours BLO - Blocked
transactionID	Corresponds to the ID of the <i>deliver_sm_resp</i> you received from SMS Gateway when you sent the original message. Used to reconcile the billing receipt with the message to which it applies.

The data is transmitted in the body of the POST as a string of key-value pairs. For example:

```
destination=tel:6175551212&serviceCode=3547&authCode=0&authMessage=&carrierName=NEXTEL  
&messageID=1m7dn6h02039u00dt444b1r61db0&outputCode=PBS&transactionID=1m7dn6h02039u00dt  
444b1r61db0
```

It's your responsibility to have a server program running at the specified URL that will accept the HTTP POST. If a receiving program isn't present, or if the POST fails for any reason, SMS Gateway will retry delivery three times over a period of 15 minutes. After three failed delivery attempts, the receipts are placed in a dead letter queue. Upon request, m-Qube will attempt to resend the queued receipts later.

Sending WAP Push Requests

To send a WAP Push via the Web Service interface, pass the `dialog.op` and `dialog.wap_url` extra parameters in a `sendMessage()` call as shown in the table below. (See [Sending Standard MT Messages](#) for the complete list of `sendMessage()` input parameters and instructions on how to format `extraParams` for SOAP and HTTP.)

Parameter	Value
serviceCode	A service code that m-Qube has specially provisioned to accept your WAP Push requests. Please contact the m-Qube Connectivity Team for more information.
message	For a WAP Push, the message parameter supplies the title of the message that appears on the handset.
extraParams	<code>dialog.op</code> Set to <code>wap_push</code> to indicate that this is a WAP Push request. <code>dialog.wap_url</code> Set to the URL being pushed. For example, <code>http://wap.yahoo.com</code> . Note that some carriers have additional restrictions on WAP pushes. See Appendix C: Carrier-Specific Notes for more information.

**SOAP example:**

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <sendMessage>
      <interpret>true</interpret>
      <destination>tel:5008885555</destination>
      <serviceCode>12345</serviceCode>
      <message>This is a WAP Push</message>
      <extraParams>
        <item>
          <key>dialog.op</key>
          <value>wap_push</value>
        </item>
        <item>
          <key>dialog.wap_url</key>
          <value>http://wap.yahoo.com</value>
        </item>
      </extraParams>
    </sendMessage>
  </soapenv:Body>
</soapenv:Envelope>
```

HTTP POST example:

```
interpret=true&destination=tel%3A5008885555&serviceCode=12345&message=This+is+a+WAP
+Push&dialog.op=wap_push&dialog.wap_url=http://wap.yahoo.com
```

Upon request, m-Qube can provision dynamic WAP Push requests that can handle complex user interactions such as content purchase and download. These push scenarios may include various kinds of automatic reminder policies for content download, including policies for re-pushing content at specific intervals or providing an alternate URL “locker” where users can access and download purchased content at a later date. Please contact the m-Qube Connectivity Team for assistance in designing and provisioning complex WAP Pushes.

Sending Parameters to m-Qube Studio

If desired, you can pass parameters to trigger an m-Qube Studio program via the SMS Gateway Web Service interface. Studio programs can add a wide range of customization to your MT messages. Please refer to the *m-Qube Studio User Guide* for a full description of the available Studio functionality.

To send parameters to Studio via the Web Service interface, call `sendMessage()` with the `serviceCode` and `extraParams` parameters set as shown in the table below. (See [Sending Standard MT Messages](#) for the complete list of `sendMessage()` input parameters and instructions on how to format extra parameters for SOAP and HTTP.)



Parameter	Value
serviceCode	The service code of the Studio program you want to trigger.
extraParams	The parameter/value pairs expected by the Studio program. For example: firstname=John lastname=Smith

SOAP example:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <sendMessage>
      <interpret>true</interpret>
      <destination>tel:5008885555</destination>
      <serviceCode>12345</serviceCode>
      <message>Hello from m-Qube</message>
      <extraParams>
        <item>
          <key>firstname</key>
          <value>John</value>
        </item>
        <item>
          <key>lastname</key>
          <value>Smith</value>
        </item>
      </extraParams>
    </sendMessage>
  </soapenv:Body>
</soapenv:Envelope>
```

HTTP POST example:

```
interpret=true&destination=tel%3A5008885555&serviceCode=12345&message=Hello+from
+m-Qube&firstname=John&lastname=Smith
```

Checking a Mobile Number

The `checkMobileNumber()` Web service method allows you to look up the wireless carrier for a given mobile phone number. `checkMobileNumber()` is callable through SOAP or via an HTTP POST or GET operation. Each method is described in detail below.

Calling `checkMobileNumber` via SOAP

For both the SOAP implementation, `checkMobileNumber()` takes the following input parameter.



Parameter	Description
mobileNumber	The subscriber's mobile phone number in UMDA format. For example: tel:5008885555 For HTTP calls, the mobileNumber value must be URL-encoded. For example: tel%3A5008885555

To call `checkMobileNumber()` via SOAP, construct your SOAP XML with the `mobileNumber` input parameter contained inside a `<checkMobileNumber>` operation tag:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <checkMobileNumber>
      <mobileNumber>tel:5008885000</mobileNumber>
    </checkMobileNumber>
  </soapenv:Body>
</soapenv:Envelope>
```

Post the XML to the following Web service endpoint:

<https://studio.mqube.us/campaignapp/services/MessagingWS>

Note that you also need to pass your m-Qube user ID and password for basic authentication.

checkMobileNumber SOAP Response

If the `checkMobileNumber()` call is successful, the Web service returns a SOAP response containing the carrier ID for the specified mobile phone number. See [Appendix A: Carrier Information](#) for the full list of carrier IDs.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <checkMobileNumberResponse
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <checkMobileNumberReturn xsi:type="xsd:string">34</checkMobileNumberReturn>
    </checkMobileNumberResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

If the phone number is not routable, the `checkMobileNumber()` call returns a value of -1.

Calling checkMobileNumber via HTTP

For the HTTP implementation, `checkMobileNumber()` takes the following input parameters.



Parameter	Description
action	Set to checkMobileNumber.
mobileNumber	The subscriber's mobile phone number in UMDA format. For example: tel:5008885555 For HTTP calls, the mobileNumber value must be URL-encoded. For example: tel%3A5008885555

To call checkMobileNumber() via HTTP, send a POST request to the following URL:

`http://studio.mqube.us/campaignapp/post/sendMessage2`

Pass the input parameters within the body of the HTTP request as a URL-encoded string. For example:

`action=checkMobileNumber&mobileNumber=tel%3A5008885555`

Note that you also need to pass your m-Qube user ID and password for basic authentication.

For interactively testing in a browser, you can also call checkMobileNumber() with an HTTP GET by passing the input parameters appended to the URL. For example:

`http://studio.mqube.us/campaignapp/post/sendMessage2?action=checkMobileNumber&mobileNumber=tel%3A5008885555`

checkMobileNumber HTTP Response

If the checkMobileNumber() call is successful, the HTTP response is an XML document containing the carrier ID of the specified mobile phone number. See [Appendix A: Carrier Information](#) for the full list of carrier IDs.

```
<checkMobileNumber>
  <carrierId>34</carrierId>
  <error />
</checkMobileNumber>
```

If the phone number is not routable, the sendMessage() call returns a carrier ID value of -1 and the HTTP response XML contains the relevant error code.

```
<checkMobileNumber>
  <carrierId>-1</carrierId>
  <error>03573- Invalid or non deliverable UMDA tel:6170000000</error>
</checkMobileNumber>
```

See [sendMessage Error Codes](#) for a list of possible errors.

Bulk Sending MT Messages

In order to efficiently handle high-volume MT message traffic, the SMS Gateway Web Service interface includes a bulk send mechanism that organizes individual messages into units of work called jobs. Each bulk send job is defined by a Job Request, which consists of a text message, optional parameters, and a list of up to a thousand recipients.

To simplify the process of creating and submitting jobs, m-Qube supplies a Bulk Send Client application that builds Job Requests from a set of input parameters, then manages all the necessary

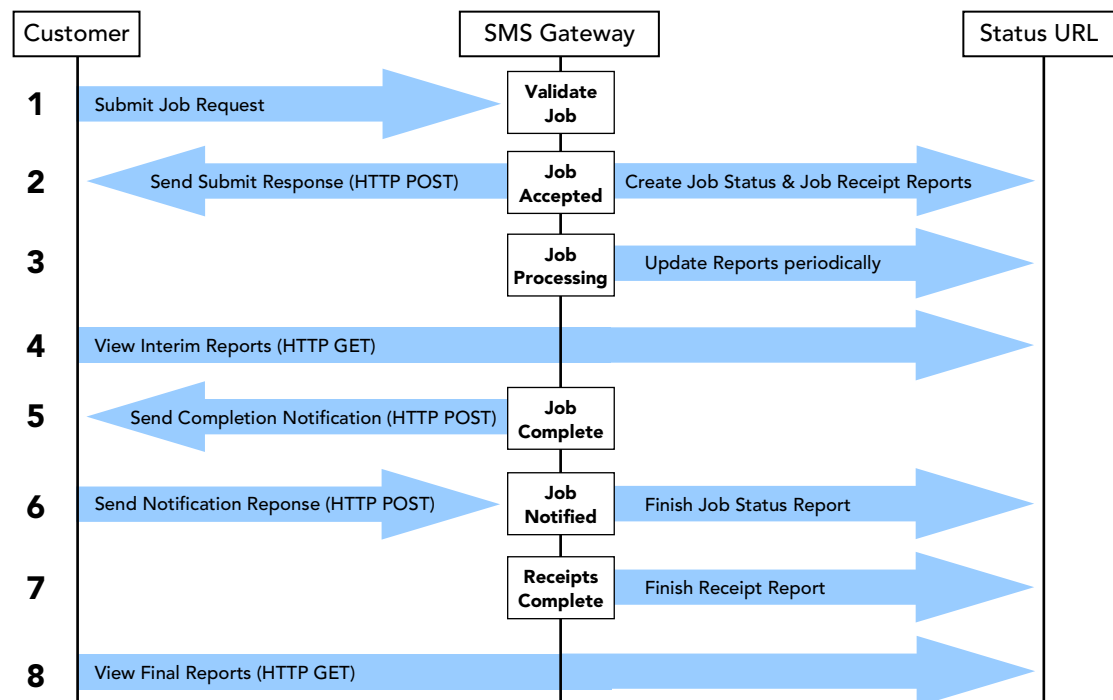


communication to and from SMS Gateway. Alternatively, you can construct your own Job Request as an XML document and submit it via HTTP POST.

To allow you to monitor the progress of a bulk send job, SMS Gateway makes available two status reports: one showing message delivery and one showing message receipts. You can view these reports using a standard Web browser, or download and parse them with a program. When the job is complete, the SMS Gateway notifies you by posting to a provided URL.

Bulk Send Job Lifecycle

This diagram illustrates the order of events in a typical bulk send job.



Step 1

You submit a Job Request to SMS Gateway through the Bulk Send Client application or via a direct HTTP post.

Step 2

SMS Gateway receives the request, validates it, and returns a submit response containing a unique URL where you can view the status reports. (See [Job Submit Response](#) for details.) Two XML status reports are available for each job: the [Job Status Report](#) tracks the success (or failure) of each message and the [Job Receipt Report](#) shows any billing or delivery receipts received in response to the messages.

Step 3

SMS Gateway begins processing the messages, updating the two reports to reflect the job's progress as it goes.

Step 4

During job processing, you can optionally monitor the reports - either using an automated program or by loading the URLs into a browser and inspecting the documents manually.



Step 5

When it determines that the job has finished, SMS Gateway marks the Job Status Report as complete and sends a notification via HTTP POST to a receiving program at a predefined URL of your choice. (See [Job Notifications](#) for more information.)

Step 6

The program receiving the notification acknowledges with an HTTP POST. After it receiving the acknowledgement, SMS Gateway marks the Job Status Report as notified.

Step 7

To give all carrier receipts time to return, SMS Gateway continues to update the Job Receipt Report for up to 72 hours after the completion of the job.

Step 8

The finished Job Status Report and Job Receipt Report are available for viewing.

Bulk Send Client Application

The easiest way to assemble and submit bulk send jobs is to use m-Qube's Bulk Send Client application. Written in Java, the Client application will construct a Job Request based on a set of input parameters then perform the necessary data transport to and from SMS Gateway - including all the required error handling and retry logic.

The Bulk Send Client is delivered as a single .jar file containing two versions:

- a Java executable intended to be invoked directly from the command line
- a Java API that allows you to embed the Client functionality in an existing application

Bulk Send Client Parameters

To start the Bulk Send Client application from the command line, use the following syntax:

```
java -jar wsgw-client.jar -u <username> -p <password> -d <destination>
-m <message> -n <numbers> -t <type> -rd <notification url>
-r SEND_FAIL|DELIVERED|BILLING_FAIL|BILLING_SUCCEED -pc <product code>
-ph <proxy host> -pp <proxy port> -log ERROR|INFO|WARN|DEBUG
```

The parameters are defined as follows:

Parameter	Name	Type	Definition
-u	Username	Required	Account username supplied by m-Qube.
-p	Password	Required	Account password supplied by m-Qube.
-d	Destination	Required	The service code you want to use for these messages.
-m	Message	Required	Path to a file containing the message body in the form of an ISO_8859-1 encoded text file.



-n	Numbers	Required	Path to an ISO_8859-1 encoded text file containing a list of Universal Mobile Device Addresses (UMDAs) to which the message should be sent. Currently, you can bulk send only to phone numbers, for which the UMDA prefix is "tel." For example: <pre>tel:6175551000</pre> Recipients should be separated by line breaks. The maximum number of recipients in a single job is currently 1000.
-t	Type	Required	Message type. Currently, only SMS is supported.
-rd	Notification URL	Required	URL to which SMS Gateway should send the completion notification. See Job Notifications for details.
-r	Receipt Options	Optional	By default, the Job Receipt Report lists only messages the carrier could not send (SEND_FAIL). You can also request one of two additional message receipt options: DELIVERED List all messages that were successfully delivered to the handset. Not all carriers support these receipts. BILLING_FAIL List only messages for which billing failed. BILLING_SUCCEED List only messages for which billing succeeded See Job Receipt Report for more information.
-pc	Product Code	Optional	Assigning a product code to a job marks the messages as premium and sets into motion the appropriate billing mechanisms in SMS Gateway. The product codes you use must be predefined with m-Qube and typically map to the standard carrier premium message price points.
-ph	Proxy Host	Optional	HTTP proxy host if required by your environment.
-pp	Proxy Port	Optional	HTTP proxy port if required by your environment.
-log	Log level	Optional	Writes an execution log to standard output. Four levels of log information are available: ERROR INFO WARN DEBUG

If there are no errors, the application prints a response message to standard output. See [Job Submit Response](#) for details.



Bulk Send Client API

If you prefer to invoke the Bulk Send Client library directly from your Java applications, the API is defined as follows:

```
// Validates, packages, and sends the bulksend data to the server for SMS processing.
public static SubmitStatus sendSMS (
    String aDestination,
    LinkedHashSet aUMDAs,
    String aMessage,
    SMSReceiptOption aReceiptOptions,
    String aUsername,
    String aPassword,
    String aJobRequestID,
    String aNotificationURL,
    String aProductCode, Map aParams)
throws IllegalArgumentException, UMDAException, MGWSCClientException

// If your network requires connection through a proxy server for HTTPS, use
// this method to configure it.
public static void setProxy(String aProxyHost, int aProxyPort)

// Generates a 28-character GUID string.
public static String getGUID()

// Returns the max number of UMDAs that will be accepted for a single call to any
// of the send methods.
public static long getJobBatchSizeMax() throws MGWSCClientException
```

The following example shows how to call the Client API from an application:

```
package com.mqube.wsgw.client;

import java.util.LinkedHashSet;
import java.util.Map;

import com.mqube.wsgw.client.enumeration.SMSReceiptOption;
import com.mqube.wsgw.enumeration.ErrorCode;

// Simple test that uses the client API to generate a job, send it to
// a server, and check the results.

public class WSGWClientTest {

    public static void main(String[] args) {

        // Get recipients from database
        LinkedHashSet UMDAs = getRecipients();

        // Set the destination (in this case, a shortcode) and
        // the message text
        String destination = "12345";
```



```
String message = "10% off all rubber duckies!";

// Provide your m-Qube username and password
String clientUser = "client1";
String clientPass = "password";

// Request receipts only for failed delivery
SMSReceiptOption receiptOption = SMSReceiptOption.SEND_FAIL;

// Notifications will be posted to the following URL
String notificationURL = "http://www.foo.com/sms/jobCompletionNotifier.php";

// getGUID() is a convenience method for generating unique IDs
String jobRequestId = WSGWClient.getGUID();

// Optionally specify an m-Qube product code
String productCode = "JOKE099";

// Set proxy setting if necessary for your network setup
// WSGWClient.setProxy("proxyServer", 8080);

Map params = null;
// Optionally set client log level
// params.put(Constants.PARAM_LOG_LEVEL, LogLevel.DEBUG);

// The WSGWClient will handle retry logic, timeouts, compression
// load balancing, and error handling.
SubmitStatus status = WSGWClient.sendSMS(destination, UMDAs, message,
    receiptOption, clientUser, clientPass, jobRequestId, notificationURL,
    productCode, params);

if (status.getStatusCode() == ErrorCode.OK) {
    // Your code can now fetch and parse the status report.
    // It will be immediately available and updated
    // periodically until the job is complete (i.e., the message
    // has been sent to all the entries in UMDAs.) It is not
    // necessary to poll the report, however, since a
    // notification will be posted to the URL specified by
    // the notificationURL provided above.
    String statusURL = status.getStatusURL();
} else {
    System.out.println("Problem submitting bulksend: "
        + status.getStatusDetails());
}

// Print the job id that the m-Qube Gateway Web Service server
// has associated with the request id.
System.out.println("Job id:" + status.getJobID() + " assigned to request "
    + jobRequestId);

// Job response details
System.out.println("Complete job response:");
```



```
        System.out.println(status);
    }

    private static LinkedHashSet getRecipients() {

        // we hard code a set of phone numbers here but more likely you will be
        // making a call to your subscriber database.
        LinkedHashSet UMDAs = new LinkedHashSet();
        // Use a Set impl that preserves order
        UMDAs.add("tel:2232123123");
        UMDAs.add("tel:2232123124");
        UMDAs.add("tel:2232123125");
        UMDAs.add("tel:2232123127");
        return UMDAs;
    }
}
```

Job Submit Response

A parser-friendly submit response is generated for each Job Request sent to the SMS Gateway. If you called the Bulk Send Client from the command line, the response is written to standard output. If you're using the Bulk Send Client API, the response is returned as a SubmitStatus object.

The response contains the following information:

job-request-id	The job request ID assigned by the Client application.
mqube-id	The m-Qube job ID added by SMS Gateway.
accepted-date	Date and time the job was accepted.
status-code	The status code for the job submission. See Job Submission Status Codes for a list of possible return codes.
status-details	A description of the status code.
max-retries	The suggested number of retries that should be attempted if the status code is an error.
retry-delay-secs	The suggested retry delay if the status code is an error.
num-umdas-accepted	The number of message recipients detected in the job.
status-url	The URL where the Job Status Report is available. The URL is a combination of your organization ID and the two job IDs: <code>http://gws.mqube.us/gws/sms/reports/<orgID>/<job-request-id>-<mqube-id>.xml</code>
receipt-url	The URL where the Job Receipt Report is available. The URL is a combination of your organization ID and the two job IDs: <code>http://gws.mqube.us/gws/sms/reports/<orgID>/<job-request-id>-<mqube-id>-receipts.xml</code>



Example Job Submit Response

```
job-request-id=abc234354659234
mqube-id=0gcpgr21r020890f1o43h1017790
accepted-date=10/21/2005 12:48:54 PM EDT
status-code=0
status-details=Job Accepted
max-retries=0
retry-delay-secs=0
num-umdas-accepted=3
status-url=http://gws.mqube.us/gws/sms/reports/7fdhts45y434908ks178m21d8641/
0gcpgr21r020890f1o43h1017790-0cbkn1j0f020890ebg43n0pih1q0.xml
receipt-url=http://gws.mqube.us/gws/sms/reports/7fdhts45y434908ks178m21d8641/
0gcpgr21r020890f1o43h1017790-0cbkn1j0f020890ebg43n0pih1q0-receipts.xml
```

Job Status Report

The Job Status Report tracks the success (or failure) of each individual message. The report is generated as soon as the job is accepted and is updated continuously during message processing. Each report has a unique URL (see [Job Submit Response](#)).

The following XML elements are returned in the report document:

<job-request-id>	The Job Request ID assigned by the Bulk Send Client application.
<mqube-id>	The m-Qube job ID added by SMS Gateway.
<recipient-count>	The number of message recipients in the job.
<status>	The first <status> element in the report indicates the status of the job as a whole. The return values are as follows: RECEIVED SMS Gateway received and accepted the job and message processing is underway. COMPLETE All messages have either been sent, failed, or retried the maximum number of times. NOTIFIED The job is complete and a Job Notification was sent and acknowledged. CANCELLED The job was stopped before completion and will not be retried.
<recipients>	The complete set of recipients.
<r>	Each individual recipient is bounded by an <r> element.
<umda>	The recipient's mobile number.
<msgid>	The message ID assigned by SMS Gateway.



<status>	The status of the individual message. Possible values are: UNSENT The message is still in the queue. SENT The message was successfully sent. RETRIABLE The message could not be sent but can be retried. This usually represents a service availability error. FAIL The message could not be sent.
<errorCode>	If the message status is FAIL, an additional element appears in the report to list the error code (when available). See Appendix B: SMS Gateway Error Codes.

Example Job Status Report

```
<job-report>
  <job-request-id>abc234354659234</job-request-id>
  <mqube-id>0gcpgr21r020890f1o43h1017790</mqube-id>
  <recipient-count>3</recipient-count>
  <status>COMPLETE</status>
  <recipients>
    <r>
      <umda>6175551000</umda>
      <msgid>0fjkjrt00g003v0b7k4351h4u2eg</msgid>
      <status>SENT</status>
    </r>
    <r>
      <umda>6175551001</umda>
      <msgid>1fsluoh00g003v0b7k4351h4u2dg</msgid>
      <status>FAIL</status>
      <errorCode>-10800</errorCode>
    </r>
    <r>
      <umda>6175551002</umda>
      <msgid>16q21pe00g003v0b7k4351h4u2a0</msgid>
      <status>SENT</status>
    </r>
  </recipients>
</job-report>
```

Job Receipt Report

The Job Receipt Report shows billing receipts received for the messages sent as part of the job. It's generated as soon as the Job Request is accepted and is updated continuously during message processing. Each report has a unique URL (see [Job Submit Response](#)).

The following XML elements are returned in the report document:



<job-request-id>	The Job Request ID assigned by the Client application.
<mqube-id>	The m-Qube job ID added by the Gateway.
<receipts>	The complete set of receipts for this job.
<receipt>	An individual message receipt.
<id>	The message ID of the receipt.
<msgid>	The ID of the original message the receipt is in response to.
<delivered>	If the message was delivered to the handset, the receipt shows a <delivered> element containing the carrier name, carrier delivery code, and carrier delivery message.
<billing-fail>	If the billing for the message failed, the receipt shows a <billing-fail> element containing the carrier name, carrier billing code, and carrier billing message.
<billing-succeed>	If the billing for the message succeeded, the receipt shows a <billing-succeed> element containing the carrier name, carrier billing code, and carrier billing message.
<send-fail>	If the message was validated and accepted by SMS Gateway but refused by the carrier, the receipt shows a <send-fail> element containing the carrier name, carrier error code, and a description of the error. Note that a send-fail receipt supersedes the SENT status given in the Job Status report, which indicates only that SMS Gateway successfully passed the message to the carrier.

NOTE

By default, the report always shows send-fail receipts. Whether or not it also includes delivered, billing-succeed, or billing-fail receipts depends on the Receipt Options you set in the Bulk Send Client parameters.

Example Job Receipt Report

```
<job-receipt-report>
  <job-request-id>abc234354659234</job-request-id>
  <mqube-id>0gcpgr21r020890f1o43h1017790</mqube-id>
  <receipts>
    <receipt>
      <id>1pr4h5100g003v0bi244410d3p50</id>
      <msgid>11o0afq00g003v0bi244410d3nvg</msgid>
      <billing-fail>carrier:Cingular;code:-10300;message:Account is not
        eligible for premium billing</billing-fail>
    </receipt>
  </receipts>
</job-receipt-report>
```



Job Notifications

When a job has finished, the SMS Gateway sends the following information via HTTP POST to the notification URL you specified in the Bulk Send Client parameters:

```
JobRequestID=id
mQubeID=id
StatusURL=url
ReceiptURL=url
```

For example:

```
JobRequestID=abc234354659234
mQubeID=0gcpgr21r020890f1o43h1017790
StatusURL=http://gws.mqube.us/gws/sms/reports/7fdhts45y434908ks178m21d8641/
0gcpgr21r020890f1o43h1017790-0cbkn1j0f020890ebg43n0pih1q0.xml
ReceiptURL=http://gws.mqube.us/gws/sms/reports/7fdhts45y434908ks178m21d8641/
0gcpgr21r020890f1o43h1017790-0cbkn1j0f020890ebg43n0pih1q0-receipts.xml
```

It's your responsibility to have a program running at the specified URL that will respond with an HTTP POST containing the two IDs in the following format:

```
JobRequestID=id
mQubeID=id
```

If SMS Gateway does not receive an appropriate response, it will re-send the notification once an hour for 24 hours.

Manually Submitting a Job Request

As an alternative to using the Bulk Send Client application to submit your jobs, you can also create your own Job Requests in XML format and submit them via HTTP POST to the WSGWXML() Web service located at the following URL:

```
http://gws.mqube.us/wsgw/WSGWXML
```

The POST must send a username and password via basic authentication and should consist of a single content parameter:

```
JobXML=xml
```

The response to the POST is the same as the Submit Response returned to the Bulk Send Client application. (See [Job Submit Response](#).)

The elements in the Job Request XML document are equivalent the parameters accepted by the Client application. (See [Bulk Send Client Parameters](#).)

Name	Definition
<job-request-id>	A unique ID that will be used to identify this job.
<destination>	The service code you want to use for these messages.
<recipient-count>	The number of message recipients included in the job. The current maximum is 1000.
<params>	A set of additional job parameters. Currently, this is used to define the notification URL (see below). However, m-Qube may add to the list of parameters as necessary.



<param>	An individual optional parameter, comprising of a <key> tag and a <val> tag.
<notification-url>	The URL to which SMS Gateway should send the completion notification. See Job Notifications for details.
<message>	The default message used for this job. You can optionally place a <message> tag inside a recipient tag to override the text used for individual messages.
<sms>	The actual text of the message.
<receipt-options>	Receipt Options for this job. See the parameter definitions in the Bulk Send Client Parameters section for details.
<product-code> (Optional)	The default product code used for this job. See the definitions in the Bulk Send Client Parameters section for details. If desired, you can place additional <product-code> tags inside individual recipient tags to override the default.
<recipients>	The complete set of recipients.
<r>	Individual recipients are bounded by an <r> element.
<umda>	The recipient's number. See the parameter definitions in the Bulk Send Client Parameters section for details.

Example Job Request XML

Note that in this example the default SMS text and product code are overridden for the last recipient.

```
<job-request>
  <job-request-id>abc234354659234</job-request-id>
  <destination>12345</destination>
  <receipt-options>BILLING_FAIL</receipt-options>
  <recipient-count>3</recipient-count>
  <params>
    <param>
      <key>job.notification.url</key>
      <val>http://sms.foo.com/sms/jobCompletionNotifier.php</val>
    </param>
  </params>
  <message>
    <sms>Two guys go into a bar...</sms>
  </message>
  <product-code>JOKE150</product-code>
  <recipients>
    <r>
      <umda>tel:6175551000</umda>
    </r>
    <r>
      <umda>tel:6175551001</umda>
    </r>
  </recipients>
</job-request>
```



```

<r>
  <umda>tel:6175551002</umda>
  <message>
    <sms>A giraffe goes into a bar...</sms>
  </message>
  <product-code>JOKE250</product-code>
</r>
</recipients>
</job-request>

```

Avoiding Duplicate Job Submissions

If you choose to send your own Job Requests via HTTP, it's critical that you also provide a mechanism to check if the job already exists before you try to resubmit it for any reason. If you resubmit a job unintentionally, you risk duplicate processing of messages already in the queue.

To see if a job is already in the queue, send an HTTP GET containing the Job Request ID to the WSGWStatus() Web service method:

```
GET http://gws.mqube.us/wsgw/WSGWStatus?JobRequestID=abc234354659221
```

If the job already exists, you'll receive a copy of the original submit response (see [Job Submit Response](#)). If the job doesn't exist, the call returns "INVALID" - which means it's safe to go ahead and submit.

If you're sending your own Job Requests, you'll also need to handle any other submission error conditions that may arise. See [Job Submission Status Codes](#) for details.

Checking Field Size Limits

Size limits for the various data fields in a Job Request are maintained remotely by m-Qube. The Bulk Send Client application loads this configuration data automatically, but if you're building your own Job Requests you need to manually retrieve the current settings by sending a simple HTTP GET to the WSGWConfig() Web service method:

```
GET http://gws.mqube.us/wsgw/WSGWConfig
```

Basic authentication is required to access the page, which returns data in key/value pair format as follows:

```

maxRecipientsPerXmlJob=1000
destinationSizeMax=18
messageSizeMax=496
urlSizeMax=1024
proxyHostSizeMax=255
proxyPortSizeMax=5
jobRequestIdSizeMax=32

```

Job Submission Status Codes

The following table lists the possible status codes returned by SMS Gateway in response to a Job Request.



Code	Error Condition	Applies To	Retry Logic
0	Job Accepted. Action has successfully completed.	Client and HTTP interface	Do not retry
1	Unauthorized. Occurs if the username or password is invalid, authentication is down, or the authentication record was removed.	Client only HTTP interface users must handle 401 response.	Stop message processing
2	Invalid argument. See the status-details field in the submit response for more information.	Client and HTTP interface	Stop message processing
3	XML parse error. See the status-details field in the submit response for more information.	HTTP interface only	Stop message processing
4	Duplicate Request ID. The request ID must be globally unique.	Client and HTTP interface	Stop message processing
5	Job processing error	Client and HTTP interface	See the max-retries field in the job submit response for the appropriate retry policy
6	Node not accepting	Client and HTTP interface	See the max-retries field in the job submit response for the appropriate retry policy
50	Connection timeout	Client only HTTP interface users must handle connection failures.	Retry every 10 minutes
51	No response from server	Client only HTTP interface users must call WSGWStatus() to see if the job was submitted	Retry every 30 seconds
52	Other network error	Client only HTTP interface users must call WSGWStatus() to see if the job was submitted	Retry every 30 seconds
55	Initialization error	Client only	Retry every 30 seconds



Receiving MO Messages

For the Web Services interface, SMS Gateway delivers MO messages via an HTTP POST to a provided URL. The body of the POST contains the following values.

Parameter	Description
dialog.message	The text of the message.
dialog.umda	The mobile phone number or e-mail address from which the message originated, in UMDA format. For example, tel:5008885555.
dialog.service_code	The service code used for the message.
dialog.carrier_id	The ID of the carrier from which the message was sent. See Appendix A: Carrier Information for the full list of carrier IDs.

Please contact the m-Qube Connectivity Team if you'd like to request additional variables for inclusion in the MO.

It's your responsibility to have a server program running at the specified URL that will accept the HTTP POST. If a receiving program isn't present, or if the POST fails for any reason, SMS Gateway will retry delivery ten times over a period of 15 minutes. After ten failed delivery attempts, the messages are placed in a dead letter queue. Since the delivery failures trigger an alarm in SMS Gateway, you will be contacted by m-Qube Support if the problem appears to lie with your server. Once the server error is fixed, m-Qube can resend the queued messages.





Appendix A: Carrier Information

This appendix lists the Tier 1 and Tier 2 North American carriers participating in the US and Canadian common short code system. For other North American connections please call m-Qube.

Note that use of advanced messaging features is dependent on carrier approval, business agreements, and provisioning by m-Qube and the participating carrier.

Carrier IDs and Direct Connection Capabilities

	ID	Content Download	Billing Method	Billing Exceptions
Tier 1 Carriers				
Alltel	7	N/A	Real-time	
Bell Mobility	8	N/A	Batch	
Boost	128	Q2 2006	Real-time	
Cincinnati Bell	125	N/A	Real-time	
Cingular Blue (AT&T Wireless)	1	WAP Push	Real-time	Prepaid Subscribers
Cingular Orange	3	WAP Push	Real-time	Prepaid or Deactivated Subscribers
Dobson	41	N/A	Real-time	
Nextel	6	WAP Push	Real-time (eWallet)	eWallet Required
Rogers Cantel	29	N/A	Batch	
Sprint PCS	5	Via text link	Real-time	
Telus Mobility	27	N/A	Batch	
T-Mobile	2	WAP Push	Real-time via Carrier Delivery Receipts	Prepaid Subscribers
US Cellular	36	N/A	Batch	
Verizon Wireless	4	Content Download API (See Note 1)	Real-time via Carrier Delivery Receipts	Deactivated or Suspended Subscribers
Virgin Mobile Canada	130	N/A	Batch	
Tier 2 Carriers				
Aliant	100	N/A	Batch	



Carolina West	145	N/A	Batch	
Cellular One of Amarillo	143	N/A	Batch	
Cellular One of San Luis Obispo	144	N/A	Batch	
Centennial	53	N/A	Batch	
Farmers Wireless	141	N/A	Batch	
First Cellular of Southern Illinois	20	N/A	Batch	
Leap Cricket	40	N/A	Batch	
MTS	127	N/A	Batch	
Peoples Wireless	147	N/A	Batch	
Pioneer/Enid	140	N/A	Batch	
RCC	35	N/A	Batch	
SaskTel Mobility	54	N/A	Batch	
TMP/Simmetry	142	N/A	Batch	
XIT	148	N/A	Batch	
Other	999	N/A	Batch	
Undeliverable	-1	N/A	Batch	

Note 1: Verizon only receives content via MMS. SMS Gateway accepts binary content via standard API and transcodes it to MMS for delivery to Verizon.

Allowable Text Message Lengths

	Technology	SMS Length	Behavior Over Limit
Tier 1 Carriers			
Alltel	CDMA/BREW	160	Not delivered
Bell Mobility	CDMA/Java	140	N/A
Boost	IDEN/Java	496	Truncated
Cincinnati Bell	GSM/TDMA	160	Truncated
Cingular Blue (AT&T Wireless)	GSM/TDMA	160 GSM 140 TDMA	Not delivered
Cingular Orange	GSM/TDMA	160 GSM 140 TDMA	Delivered as multiple messages
Dobson	GSM/TDMA	160	Delivered as multiple messages



Nextel	IDEN/Java	500	See Appendix C: Carrier-Specific Notes .
Rogers Cantel	GSM/Java	160 GSM 140 TDMA	N/A
Sprint PCS	CDMA/Java	160	Truncated
Telus Mobility	CDMA/Java	160	Delivered as multiple messages
T-Mobile	GSM/Java	140	Not delivered
US Cellular	CDMA/BREW	140	Truncated
Verizon Wireless	CDMA/BREW	160	Truncated
Virgin Mobile Canada	CDMA/Java	140	Truncated

Tier 2 Carriers

Aliant	CDMA	140	N/A
Carolina West	CDMA	140	Delivered as multiple messages
Cellular One of Amarillo	GSM	160	N/A
Cellular One of San Luis Obispo	GSM	150	N/A
Centennial	GSM	160	N/A
Farmers Wireless	GSM	140	Delivered as multiple messages
First Cellular of Southern Illinois	GSM/CDMA	160	Delivered as multiple messages
Leap Cricket	CDMA	160	N/A
MTS	CDMA	200	N/A
Peoples Wireless	CDMA	160	N/A
Pioneer/Enid	CDMA/TDMA	160	Delivered as multiple messages
RCC	CDMA	160	N/A
SaskTel Mobility	CDMA	160	N/A
TMP/Simmetry	GSM	160	N/A
XIT	GSM	160	N/A



Appendix B: SMS Gateway Error Codes

The following table lists the error codes returned from the m-Qube SMS Gateway. For the SMPP interface, the error codes appear in the message receipt. For the Web Services interface, the error codes are returned in the SOAP or HTTP response (for individual MT messages) or are listed in the Job Status and Job Receipt Reports (for bulk send messages).

Code	Description	Type	Explanation	Source
(SMPP) NACK 11	Recurring Ineligible Numbers	Fatal	This error code is returned in cases where the m-Qube Gateway cannot deliver a message to the given subscriber MDN. These errors are reported back in real time and do not appear on billing reports.	m-Qube
-1	m-Qube Internal Processing Error	Severe	An m-Qube internal processing error has occurred.	m-Qube
1	Enroute	Retriable	SMPP Carrier Delivery Receipt: The message is in Enroute State.	Virgin Mobile Bell Mobility Aliant
3	Carrier Network Delivery Failure	Retriable	SMPP Carrier Delivery Receipt: Validity Period Expired. Carriers that support delivery receipts, such as Verizon, receive this error when m-Qube has not heard back from the carrier for a delivery receipt after a period of time.	SMPP
4	Message Has Been Deleted	No Action	SMPP Carrier Delivery Receipt: The end user has deleted the message.	SMPP
5	Delivery Failure	Severe	SMPP Carrier Delivery Receipt: The message failed to reach the subscriber's handset.	SMPP
6	Message in Accepted State	No Action	SMPP Carrier Delivery Receipt: The message was successfully sent.	SMPP
7	Message in Invalid State	Fatal	SMPP Carrier Delivery Receipt: The message was invalid. This occurs when you attempt to send content to a handset which does not support that format.	SMPP



8	Message in Rejected State	Severe	SMPP Carrier Delivery Receipt: The message was rejected because the subscriber is no longer active or is ineligible for premium services.	SMPP
14	Overlong Message	Severe	SMPP Carrier Delivery Receipt: The message was too long and could not be delivered.	SMPP
23	Invalid Originator Address	Severe	SMPP Carrier Delivery Receipt: The message was not delivered because the short code was not correctly provisioned on the carrier.	SMPP
24	Invalid Destination Address	Severe	SMPP Carrier Delivery Receipt: The message was not delivered because the phone number does not appear in the carrier's subscriber database.	SMPP
35	Prepaid Balance Exceeded	Retriable	A Verizon prepaid subscriber has an insufficient balance for this transaction.	SMPP for Verizon
68	Message Queue Full	Retriable	See error 63490.	Verizon
74	Submit Failure	Severe	SMPP Carrier Delivery Receipt: The message failed to reach the subscriber's handset.	Verizon
99	General System Error	Severe	Verizon protocol or system availability error.	SMPP for Verizon
-100	Miscellaneous Error	Severe	See error 63490.	Cingular Orange
-101	Authentication Failed	Severe	System configuration error. Check with m-Qube.	Sprint
-102	Unsupported Version	Severe	System configuration error. Check with m-Qube.	Sprint
-103	Internal Error	Retriable	Retry. If not successful, check with m-Qube.	Sprint
-104	Sprint System Error	Retriable	Retry. If not successful, check with m-Qube.	Sprint
-105	Non-Sprint MIN	Fatal	The MDN was not found in Sprint's billing system.	Sprint
-106	Incorrect Reseller ID for MIN	Severe	System configuration error. Check with m-Qube.	Sprint



-107	Invalid MIN Format	Severe	System configuration error. Check with m-Qube.	Sprint
-108	Unsupported Reseller ID	Severe	System configuration error. Check with m-Qube.	Sprint
-109	Invalid Product ID	Severe	System configuration error. Check with m-Qube.	Sprint
-110	Invalid Short Code	Fatal	The short code is not live. Check with m-Qube.	Sprint
-111	Blocked MDN	Fatal	The MDN is no longer active and should be removed from any subscriptions.	Sprint
-112	Malformed XML	Severe	Check parameters before retry.	Sprint
-113	Invalid Charge ID	Severe	System configuration error. Check with m-Qube.	Sprint
-10100	Account is Locked (Spending Limit Reached)	Retriable	See error 40040.	Cingular Blue
-10200	Service Not Available	Retriable	See error 63100 and 63499.	Cingular Blue
-10300	Account Is Not Eligible for Premium Billing	Fatal	This code is returned if the customer is not eligible to have charges billed to the phone (for example, pre-paid subscribers) or if the default billing method is via credit card.	Cingular Blue
-10400	Account is Closed	Fatal	See error 40020.	Previously Cingular Orange
-10500	m-Qube Premium Configuration Error: Invalid Product ID	Severe	See error 62010.	Cingular Blue
-10600	Account is Locked	Retriable	There is a problem with the customer's account. A possible trigger for this status code is a suspended account. In the case of Cingular, this can also mean the MIN is an ineligible Prepaid TracFone Subscriber. Cingular no longer permits Prepaid TracFone subscribers to participate.	Cingular Blue



-10700	User Login Failed	Fatal	See error 40010.	Previously Cingular Orange
-10800	User Not Found	Fatal	See error 40010.	Previously Cingular Orange
-20100	Account Not Found	Fatal	The user's phone number does not exist in the system. Prior to December 2005, a high volume of -20100 errors were returned by Cingular Blue because m-Qube did not have access to the necessary porting information in the wake of the Blue-Orange merger. At the end of November 2005, m-Qube implemented a new solution that resolves ports via number look-up. Going forward, these errors should mainly be due to inactive MDNs.	Cingular Blue
-20199	m-Qube Premium Configuration Error	Severe	Retail Price does not match Carrier Retail Price.	m-Qube
-20200	Premium Transaction Time Limit Has Been Exceeded	Retriable	This error has two primary sources: 1) On Verizon Wireless it means that m-Qube did not receive a message Delivery Report within 120 hours. 2) On other carriers, any outage or lack of carrier response (billing or messaging) that prevents m-Qube from delivering and billing for a message will result in the message being queued and retried. After 120 hours have passed if the retries were unsuccessful, the transaction fails and generates this error code.	m-Qube
-20201	Undeliverable	Severe	This is an m-Qube internal processing error.	m-Qube



-20202	Subscriber's Carrier is Not Participating in this Premium Program	Severe	This will occur whenever a carrier does not support a program. Some common examples of this are: 1) Carrier is not participating in specific short code. In some cases, short codes do not leverage the entire network of carriers for MMA compliance reasons as decided by carriers and customers. In these cases, if a subscriber from a "non-participating" carrier attempts to participate, m-Qube fails the transaction. 2) Carrier is not enabled on binary content. Not all carriers in the m-Qube network are enabled with Binary content. Examples as of 09/05 include Sprint, VzW (will be enabled in Q4), and Dobson. As such, if subscribers for one of these carriers attempt to purchase binary content, they will be failed with this corresponding error. 3) Carrier is not yet live. In some cases, m-Qube is in the process of launching new carriers, but has not yet turned them on. If subscribers from one of those "non-participating" carriers attempt to purchase content, they will receive this error.	m-Qube
-20300	Not Supported	Fatal for T-Mobile Retriable for all other carriers	The request contained an attribute or value that is not supported by the Purchase Service. If additional information is available, this will be added to the Result Message returned in the SOAP response. This error mainly originates from T-Mobile, where premium transactions are not allowed for prepaid customers.	All Carriers
-20301	Delivery Failure	Severe/ Fatal	T-Mobile requires m-Qube to validate that the message was successfully delivered as a precondition to submitting to T-Mobile for billing. This error likely means the subscriber is no longer active.	T-Mobile
-20501	Miscellaneous Protocol Error	Severe/ Fatal	The T-Mobile system is unavailable.	T-Mobile



-20600	VzW \$100 Subscription Limit	Retriable	m-Qube implements a \$100 per month, per program limit for Verizon Wireless subscribers. This error message is generated when a given subscriber hits the \$100 limit.	m-Qube for VzW
-20601	m-Qube Internal Processing Error	Severe	m-Qube internal processing error.	m-Qube
40010	User Not Found	Severe for Nextel & Boost Fatal for all other carriers	This status code is returned if the subscriber was not found in the system and the Purchase Service does not provision a user for this domain. This status code is also returned if the client certificate contains an incorrect domain. Nextel and Boost subscribers often generate this error due to the eWallet user experience. Because they have so many corporate accounts, Nextel and Boost don't allow subscribers to bill premium services to the phone. Instead, subscribers have to use eWallet as their payment device. As a result, when they attempt to purchase a premium service for the first time, they're taken through a multi-step eWallet registration flow on their phone that requires them to provide a credit card number. This flow has a high drop-off rate, and each subscriber who abandons the process causes a 40010 error.	Cingular Orange Nextel Boost Dobson Alltel
40020	Account is Closed	Fatal	Customer's account has been closed.	Cingular Orange Nextel Boost Dobson Alltel
40030	Account is Locked	Retriable or Fatal	There is a problem with the customer's account. A possible trigger for this status code is a suspended account.	Cingular Orange Nextel Boost Dobson Alltel
40040	Over Spending Limit	Retriable	The customer has reached his/her predetermined spending limit.	Cingular Orange Nextel Boost Dobson Alltel



40050	Credit Card Previously Invalidated	Severe	An eWallet customer is using an invalid credit card number.	Cingular Orange Nextel Boost Dobson Alltel
40060	Credit Card Invalid	Severe	An eWallet customer is using an invalid credit card number.	Cingular Orange Nextel Boost Dobson Alltel
42010	Ineligible Product	Fatal	This code is returned if any of the following is true: - The customer is not eligible for having charges billed to the phone. - The product is not eligible for billed to phone purchasing. - The product has an incorrect authorization level.	Cingular Orange Nextel Boost Dobson Alltel
62010	Invalid Product ID	Retriable	No such product exists in the product catalog.	Cingular Orange Nextel Boost Dobson Alltel
63010	Authentication Failure	Retriable	Could not authenticate the caller (i.e., Content Provider) specified in the Request message (typically based on Client Certificates).	Cingular Orange Nextel Boost Dobson Alltel
63100	Service Not Available	Retriable	Communication with an external system failed. This could mean that the network operator's system is down or it could mean that a cryptography error occurred during communication between the Purchase Service and the network operator's system. If the latter is true, the likely cause is a configuration problem on one side or the other.	Cingular Orange Nextel Boost Dobson Alltel



63490	Not Supported	Fatal	The request contained an attribute or value that is not currently supported by the Purchase Service. If additional information is available, it will be added to the Response Text. Historical failure reasons include: 1) cingMobileMAGID (Mobile Access Gateway ID) not found for userId: 4052034261 from premium authorization. This occurs if the subscriber: - is a Blue or Orange subscriber who has never used his or her WAP browser. According to Cingular, this gets fixed after 72 hours after first use of the WAP browser. - is a new Cingular/MediaNet subscriber. According to Cingular, this gets fixed approximately 72 hours after subscriber activation. - is a TDMA customer without WAP capability. 2) Unsupported prepaid platform value: TracFone from premium authorization. The transaction cannot be retried (i.e., it will fail repeatedly with the same error.) 3.) Locality(L) not found in search results: from premium authorization. This does not appear to be a recoverable error.	Cingular Orange Nextel Boost Dobson Alltel
63499	General System Error	Retriable	A system error occurred. If additional information is available, it will be added to the Response Text. The text provided with the message relates to credit card authorization being rejected.	Cingular Orange Nextel Boost Dobson Alltel
-100000	VzW \$100 Subscription Limit	Retriable	m-Qube implements a \$100 per month, per program limit for Verizon Wireless subscribers. This error message is generated when a given subscriber hits the \$100 limit.	m-Qube for VzW



Appendix C: Carrier-Specific Notes

Nextel Message Length

Nextel's messaging implementation uses up to 500 characters instead of the 160 or so characters normally allowed by SMS. If the text of an SMS message sent via SMPP exceeds 250 characters, the message text is encoded differently within the *deliver_sm* PDU. Instead of appearing in the *short_message* field, the text is placed in the *message_payload* optional field (see the SMPP specification for details). Before passing along an MO message, SMS Gateway moves the message text back into the *short_message* field - which imposes a de facto 250-character limit. If you need to configure SMS Gateway to pass more than 250 characters, please contact the m-Qube Connectivity Team.

Also, Nextel messages often begin with a "Re: (subject)" string and sometimes append "In Reply..." followed by the original message. By default, SMS Gateway is configured to strip out this additional text. If you need it retained, please contact the m-Qube Connectivity Team.

Sprint WAP Push URLs

Sprint only allows its users to download content from preapproved URLs. If you attempt to send a WAP push that uses a non-recognized location, the download is blocked and the recipients see an "Untrusted Content Error" on their handset. In order to avoid this error, you'll need to ask Sprint to set up a whitelisted DNS alias that maps to your actual content URL. If you need help with this process, please contact the m-Qube Connectivity Team.





Appendix D: Technical Support and Sales

m-Qube Technical Support

If you have questions that are not answered by this user guide or encounter technical problems with SMS Gateway, please contact m-Qube via the Client Support System at the following address:

<http://www.m-qube.com/clientsupport>

If you do not have access to the Client Support System, please contact your m-Qube account manager for assistance.

m-Qube Sales and Professional Services

m-Qube has significant experience with a wide range of mobile marketing programs. You can contact m-Qube Sales and Professional Services at sales@m-qube.com.