GLOBAL MOBILE MESSAGING

Technical Specification

Q-CASTER 6.0
TO THE QUIOS MESSAGING PLATFORM

Monday, August 13, 2007

support@quios.net

Table of Contents

1 Introduction

Authorized users of the Q-Caster 6.0 services can programmatically submit XML messages through a simple SOAP interface. These messages are converted to SMS and distributed to mobile Handsets worldwide. Q-Caster is ideal for sending large numbers of messages directly from a database or other content provider application.

Q-Caster 6.0 accepts two types of submissions: single and multiple. Use the single submission mode, `send_to_number`, to submit one message to one Handset. Use the multiple mode, `send_to_numbers`, to submit one message to multiple Handsets.

Preliminary results of the message are available immediately, and indicate the initial validation and error check that Q-Caster performs before sending the message to any downstream providers. Further delivery information is also available, up to and including final receipt by the Handset.

2 Prerequisites

To submit messages to Q-Caster for transmission, the following conditions must be met:

- Provider must have a current, valid Contract for Message Distribution Services with Quios.
- Provider must have an existing Quios account with valid authentication information (username and password).
- At least one valid static IP number or range (CIDR) must be associated with Provider's username/password.
- Provider must have a Calling Application capable of transmitting and receiving SOAP requests and responses in accordance with these specifications.
- Provider must have a valid access number for each Handset intended as a Message destination, including necessary country codes and area codes.

This technical specifications document assumes familiarity with the following standards, protocols, specifications, and RFCs:

- RFC 1738 “Uniform Resource Locators (URL)”
- W3C Note “Simple Object Access Protocol (SOAP) 1.1”
- W3C Note "Web Services Description Language (WSDL) 1.1"
- "SMTP Transport Binding for SOAP 1.1"
- ETSI GSM 03.38 Specification
- The Unicode Standard
- Nokia Smart Messaging Spec
- Nokia Smart Messaging FAQ
- WAP Service Indication Specification
- WAP Service Loading Specification

3 System elements and terminology

In addition to industry-standard terminology, this document defines additional terms as listed in Table 3-1.

Table -1 Terminology used in Q-Caster Technical Specifications

Calling Application	The programmatic interface that produces the Provider's message and receives resultant success/error notifications.
Request	The complete SOAP request transmitted to Q-Caster.
Response	The complete SOAP response transmitted from Q-Caster.
Message	The content (text, ringtone, etc.) intended for a Handset. Long Messages are divided into multiple SMSs.
SMS	A single SMS to be delivered to a single Handset. Can be text, ringtone, etc. Can be a portion of a long Message that was too large to deliver as a single SMS.
Handset	Mobile telephone, text pager, or other device capable of receiving SMS messages.
Provider	The organization that provides the Request and sends it to Q-Caster for distribution to the Handsets. Quios also uses the services of "downstream providers".
Consumer	The end-user to whose Handset the Provider is sending Messages.

4 Connecting to the Q-Caster server

In order to transmit the Request to the Q-Caster system for distribution, the Calling Application must establish a network (TCP/IP) connection with the Q-Caster access server.

Connections must always be made from an IP address that is registered with Quios as an authorized address for the Account. Connection attempts from unauthorized locations are rejected.

4.1 Connecting through HTTP

To transmit Requests, the Provider directs the Calling Application to connect to the following URL:

```
http://soap.ewingz.com/SOAP/
```

It is recommended that the connection always be made to the symbolic DNS name for the access server rather than to the IP address. The IP address associated with the server name is subject to change without notice.

4.2 Support for legacy methods

Qcaster 6.0 currently does not support the methods described in paragraphs 10 and 11 (`cancel`, `number_lookup`, `carrier_list`, `coverage_map`).

To transmit Requests using these legacy methods, the Provider needs to direct the Calling Application to connect to the Qcaster URL:

```
http://soap.ewingz.com/SOAP/
```

and use the **Q-Caster 4.0** namespace URI reference:

```
http://soap.ewingz.com/eWingz/SOAP/QC40
```

It is recommended that the connection always be made to the symbolic DNS name for the access server rather than to the IP address. The IP address associated with the server name is subject to change without notice.

5 Sending messages

Requests are transmitted to Q-Caster as a SOAP RPC request. The interface for composing the request is the responsibility of the Calling Application. See the W3C Note "Simple Object Access Protocol (SOAP) 1.1" for more details on using SOAP. The Q-Caster namespace URI reference is:

```
http://soap.ewingz.com/eWingz/SOAP/QC60
```

To use the Q-Caster service, the Calling Application must be able to make a SOAP request and read the responses to it. Most applications will make use of an XML parser and/or SOAP toolkit.

Note: The order in which the parameters are passed to the Q-Caster server is very important. If any of the parameters are passed out of order, the request will fail.

5.1 The `send_to_number` RPC and the `send_to_numbers` RPC

To send one or more SMS messages, the Calling Application submits a SOAP request containing a call to `send_to_number` or `send_to_numbers`. The `send_to_number` method sends a single message to a single Handset. The `send_to_numbers` method sends a single message to multiple Handsets. Most parameters of the latter request are similar to the `send_to_number` request, except that `send_to_numbers` groups the `msisdn`, `uniqueid`, `set_reply_path`, and `deliver_after` into an array. Using this array, the Calling Application can send the message to multiple Handsets.

The parameters for these methods are listed in tables 5-1 and 5-2. Unless where mentioned, Q-Caster does not have default values for these parameters.

Table 5-1 Parameters to `send_to_number` request

Parameter	Type	Constraints	Meaning
`username`	string	must be valid username for the submitting IP address	Provider's username for authentication.
`password`	string	must be valid password for this username	A valid password for the username; used for authentication.
`testmode`	boolean	[true | false]	`testmode=true` indicates that the request is in test mode, which performs all authentication, validation, and parsing steps but does not deliver SMSs to Handsets nor debits Provider accounts.
`notification`	string	[none | quios | handset]	Sets the extent of delivery information available in regard to this Message. See Section 7.2 for details.
`type`	string	[GSM0338 | UCS2 | Binary | RTTTL]	Indicates the type of information contained in the Message. See Section 6 for more information.
`class`	integer	[0 | 1 | 2 | 3]	Indicates the SMS class of GSM0338 messages; 0 is a flash message, 1 is delivered to memory, 3 is delivered to SIM, and 4 is delivered to serial port. However, options 3 and 4 are not supported by all handsets and their use is not recommended. Class is used only when `type= GSM0338` or `UCS2`. If `type` is `Binary` or `RTTTL`, then this field is ignored.
`udhi`	boolean	[true | false]	`udhi=true` indicates that the `header`+`body`+`footer` of this Message contain a UDHI-compliant message.
`originator`	base64 Binary	see Section 6.9 and 6.10	Sets the SMS originator to the specified string dynamically. Defaults to the `originator` string that is associated with this Provider. If the value of the `originator` string is zero length, then the Message is delivered with the default `originator`. See Section 6.9 and 6.10 for more details.
`header`	base64 Binary	limited to 80 bytes	Contains the message header. Normally `header`, `body`, and `footer` are concatenated together for display on the Handset. However, if `body` is an array, then `header` is ignored. This Quios `header` parameter is not the same as the UDHI header.
`body`	base64 Binary or array of	limited to 4000 bytes; `header`+ `body`+`footer` must be >0.	Contains the binary data or text to be transmitted to the Handset. This Quios `body` parameter is not the same as the UDHI body.

Parameter	Type	Constraints	Meaning
	base64 Binary		
footer	base64 Binary	limited to 80 bytes	Contains the message footer. Normally header, body, and footer are concatenated together for display on the Handset. However, if body is an array, then footer is ignored.
msisdn	string	minimum 7 digits, maximum 15 digits. International format. No spaces or alpha characters allowed. The + character is not allowed.	Indicates the MSISDN (phone number) of the Handset.
uniqueid	base64 Binary	unless zero-length, must be unique over the entire lifetime of the account, and limited to 80 bytes	A unique string to identify the Message; multiple SMSs can have the same uniqueid if they were split automatically. Used to reference Messages for checking delivery status. Q-Caster assigns a value for uniqueid to any Message with a zero-length uniqueid.
set_reply_path	Boolean	[true|false]	set_reply_path=true indicates the reply path should be set. set_reply_path=false indicates that the reply path should not be set.
deliver_after	date	format is YYYY-MM-DD HH:MM:SS	Indicates the date and time (in GMT) at which the message is to leave the Q-Caster system for delivery by downstream providers.
price	String	Price is specified in cents ($1.99 is specified as '199')	Price at which the message needs to be billed on the cell phone bill of the consumer. 'FTEU' indicates 'free-to-enduser' on Cingular Orange only.
campaign_id	Integer		ID used to identify each one of the Provider's campaigns. Contact Quios to find out the campaign id values.
quios_sub	String	start|stop|cancel|refund	The action that needs to be taken on a subscription. If NULL, the message is considered to be not subscription related.
quios_sub_id	Base64 Binary		A unique string to identify the subscription, which corresponds to the uniqueid of the message that started the subscription.
content_type	String	Max 80 bytes	Specifies what type of content is sent with this text message. See 6.12 for details.

Table 5-2 Parameters to `send_to_numbers` request

Parameter	Type	Constraints	Meaning
`username`	string	must be valid username for the submitting IP address	Provider's username for authentication.
`password`	string	must be valid password for this username	A valid password for the username; used for authentication.
`testmode`	boolean	[true | false]	`testmode=true` indicates that the Rrequest is in test mode, which performs all authentication, validation, and parsing steps but does not deliver SMSs to Handsets nor debits Provider accounts.
`notification`	string	[none | quios | handset]	Sets the extent of delivery information available in regard to this Message. See Section 7.2 for details.
`type`	string	[GSM0338 | UCS2 | Binary | RTTTL]	Indicates the type of information contained in the Message. See Section 6 for more information.
`class`	integer	[0 | 1 | 2 | 3]	Indicates the SMS class of GSM0338 messages; 0 is a flash message, 1 is delivered to memory, 3 is delivered to SIM, and 4 is delivered to serial port. However, options 3 and 4 are not supported by all handsets and their use is not recommended. Class is used only when `type= GSM0338` or `UCS2`. If `type` is `Binary` or `RTTTL`, then this field is ignored.
`udhi`	boolean	[true | false]	`udhi=true` indicates that the `header`+`body`+`footer` of this Message contain a UDHI-compliant message.
`originator`	base64 Binary	see Section 6.9 and 6.10	Sets the SMS originator to the specified string dynamically. Defaults to the `originator` string that is associated with this Provider. If the value of the `originator` string is zero length, then the Message is delivered with the default `originator`. See Section 6.9 and 6.10 for more details.
`header`	base64 Binary	limited to 80 bytes	Contains the message header. Normally `header`, `body`, and `footer` are concatenated together for display on the Handset. However, if `body` is an array, then `header` is ignored. This Quios `header` parameter is not the same as the UDHI header.
`body`	base64 Binary or array of	limited to 4000 bytes; `header`+ `body`+`footer` must be >0.	Contains the binary data or text to be transmitted to the Handset. This Quios `body` parameter is not the same as the UDHI body.

Parameter	Type	Constraints	Meaning
	base64 Binary		
footer	base64 Binary	limited to 80 bytes	Contains the message footer. Normally header, body, and footer are concatenated together for display on the Handset. However, if body is an array, then footer is ignored.
price	String	Price is specified in cents ($1.99 is specified as '199')	Price at which the message needs to be billed on the cell phone bill of the consumer. 'FTEU' indicates 'free-to-enduser' on Cingular Orange only.
campaign_id	Integer		ID used to identify each one of the Provider's campaigns. Contact Quios to find out the campaign id values.
numbers	struct (array)	Consists of msisdn, unique_id, set_reply_path, and deliver_after.	See send_to_number method for detailed description of the elements of the array.
quios_sub	String	start|stop|cancel|refund	The action that needs to be taken on a subscription. If NULL, the message is considered to be not subscription related.
quios_sub_id	Base64 Binary	Max. 80 bytes	A unique string to identify the subscription, which corresponds to the uniqueid of the message that started the subscription.
content_type	String		Specifies what type of content is sent with this text message. See 6.12 for details.

5.2 WAP Push messages

WAP Push messages provide a way to indicate and/or execute services on a Handset by pointing the Handset at a content URL. The service can be MMS messages, Java applications, WML files, etc. For example, the Handset might be notified of new e-mail messages, and provided with a URL from which to retrieve those messages.

The Quios WAP Push feature offers a convenient interface for sending WAP Push messages, eliminating the need for the Calling Application to perform the binary encoding for such messages.

There are three types of WAP Push messages. A Service Indication (SI) displays a message on the Handset and prompts the user to access the service; this feature is accessed using the wap_push_si request. Service Loading (SL) loads the service on the Handset without any prompt to the user; it is accessed through the wap_push_sl request.

For maximal compatibility with downstream providers, Quios recommends that wap_push_si contain both href and text, while wap_push_sl is limited to href only. Other optional parameters provided in the WAP specifications are unpredictable on many networks and Handsets.

The third WAP type, Cache Operation (CO), is not implemented in Q-Caster.

Table 5-2 Parameters to `wap_push_si` request

Parameter	Type	Constraints	Meaning
`username`	string	Must be valid username for the submitting IP address	Provider's username for authentication. This parameter is unchanged from QC2.5.
`password`	string	Must be valid password for this username	A valid password for the username; used for authentication. This parameter is unchanged from QC2.5.
`testmode`	boolean	[true | false]	`testmode=true` indicates that the request is in test mode, which performs all authentication, validation, and parsing steps but does not deliver SMSs to Handsets nor debits Provider accounts. Equivalent to test_mode in QC2.5.
`notification`	string	[none | quios | handset]	Sets the extent of delivery information available in regard to this Message. See Section 7.2 for details.
`originator`	base64 Binary	see Section 6.9 and 6.10	Sets the SMS originator to the specified string dynamically. Defaults to the `originator` string that is associated with this Provider. If the value of the `originator` string is zero length, then the Message is delivered with the default `originator`. See Section 6.9 and 6.10 for more details.
`msisdn`	string	minimum 7 digits, maximum 15 digits. International format. No spaces or alpha characters allowed. The + character is not allowed.	Indicates the MSISDN (phone number) of the Handset.
`uniqueid`	base64 Binary	unless zero-length, must be unique over the entire lifetime of the account, and limited to 80 bytes	A unique string to identify the Message; multiple SMSs can have the same `uniqueid` if they were split automatically. Used to reference Messages for checking delivery status. Q-Caster assigns a value for `uniqueid` to any Message with a zero-length uniqueid.
`deliver_after`	date	format is YYYY-MM-DD HH:MM:SS	Indicates the date and time (in GMT) at which the message is to leave the Q-Caster system for delivery by downstream providers.
`price`	String	Price is	Price at which the message needs to

Parameter	Type	Constraints	Meaning
		specified in cents ($1.99 is specified as '199')	be billed on the cell phone bill of the consumer. 'FTEU' indicates 'free-to-enduser' on Cingular Orange only.
`campaign_id`	Integer		ID used to identify each one of the Provider's campaigns. Contact Quios to find out the campaign id values.
`quios_sub`	String	start|stop|cancel|refund	The action that needs to be taken on a subscription. If NULL, the message is considered to be not subscription related.
`quios_sub_id`	Base64 Binary	Max. 80 bytes	A unique string to identify the subscription, which corresponds to the `uniqueid` of the message that started the subscription.
`content_type`	String		Specifies what type of content is sent with this text message. See 6.12 for details.
`href`	string	Must be valid HTTP URL	Specifies the location of the content to which the Handset is directed.
`action`	string	[signal-none | signal-low | signal-medium | signal-high | delete]	Optional; indicates how the Handset acts on the SI; default is `signal-medium`. `delete` indicates that the current SI and any SI with the same `si_id` should be automatically deleted from the Handset. `signal-none` uses the unimplemented info element to determine action. The other signal-levels indicate the level of user-intrusiveness that the delivery will effect; implementation is determined by the Handset but might include volume adjustments, sound versus vibrate, etc.
`created`	string	format YYYY-MM-DDThh:mm:ssZ	Optional; indicates date/time in GMT/UTC that the content (not the SI) was created or last modified.
`si_expires`	string	format YYYY-MM-DDThh:mm:ssZ	Optional, indicates date/time in GMT/UTC that the message expires from the Handset.
`si_id`	string		Optional; provides an identifier for distinguishing between different SIs; if blank, the value of `href` is used.
`text`	string		Optional (but recommended); contents are displayed on the Handset upon delivery.

Table 5-3 Parameters to `wap_push_sl` request

Parameter	Type	Constraints	Meaning
`username`	string	must be valid username for the submitting IP address	Provider's username for authentication.
`password`	string	must be valid password for this username	A valid password for the username; used for authentication. This parameter is unchanged from QC2.5.
`testmode`	boolean	[true | false]	`testmode=true` indicates that the request is in test mode, which performs all authentication, validation, and parsing steps but does not deliver SMSs to Handsets nor debits Provider accounts.
`notification`	string	[none | quios | handset]	Sets the extent of delivery information available in regard to this Message. See Section 7.2 for details.
`originator`	base64 Binary	see Section 6.9 and 6.10	Sets the SMS originator to the specified string dynamically. Defaults to the `originator` string that is associated with this Provider. If the value of the `originator` string is zero length, then the Message is delivered with the default `originator`. See Section 6.9 and 6.10 for more details.
`msisdn`	string	minimum 7 digits, maximum 15 digits. International format. No spaces or alpha characters allowed. The + character is not allowed.	Indicates the MSISDN (phone number) of the Handset.
`uniqueid`	base64 Binary	unless zero-length, must be unique over the entire lifetime of the account, and limited to 80 bytes	A unique string to identify the Message; multiple SMSs can have the same uniqueid if they were split automatically. Used to reference Messages for checking delivery status. Q-Caster assigns a value for uniqueid to any Message with a zero-length uniqueid.
`deliver_after`	date	format is YYYY-MM-DD HH:MM:SS	Indicates the date and time (in GMT) at which the message is to leave the Q-Caster system for delivery by downstream providers.
`price`	String	Price is specified in cents ($1.99 is specified as	Price at which the message needs to be billed on the cell phone bill of the consumer. 'FTEU' indicates 'free-to-enduser' on Cingular Orange only.

Parameter	Type	Constraints	Meaning
		'199')	
campaign_id	Integer		ID used to identify each one of the Provider's campaigns. Contact Quios to find out the campaign id values.
quios_sub	String	start|stop|cancel|refund	The action that needs to be taken on a subscription. If NULL, the message is considered to be not subscription related.
quios_sub_id	Base64 Binary	Max. 80 bytes	A unique string to identify the subscription, which corresponds to the uniqueid of the message that started the subscription.
content_type	String		Specifies what type of content is sent with this text message. See 6.12 for details.
href	string	must be valid HTTP URI	Specifies the location of the content to which the Handset is directed.
action	string	[execute-low | execute-high | cache]	Optional, indicates the urgency with which the Handset accesses the content. execute-low (the default) loads content in the same way as the Handset loads user-initiated requests. execute-high is similar, but might result in user-intrusive behavior. cache loads the service, but places it into the cache instead of executing. If no cache is available, the SL is silently discarded.

Listing 5-1 Example `send_to_number` request

```
 <?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <namesp1:send_to_number xmlns:namesp1="http://soap.ewingz.com/eWingz/SOAP/Q
C60">
      <username xsi:type="xsd:string">xxxxxxx</username>
      <password xsi:type="xsd:string">xxxxxxx</password>
      <testmode xsi:type="xsd:boolean">false</testmode>
      <notification xsi:type="xsd:string">handset</notification>
      <type xsi:type="xsd:string">GSM0338</type>
      <class xsi:type="xsd:integer">1</class>
      <udhi xsi:type="xsd:boolean">false</udhi>
      <originator xsi:type="xsd:base64Binary">xxxxx=</originator>
      <header xsi:type="xsd:base64Binary" />
      <body xsi:type="xsd:base64Binary">VmlzaXQgUXVpb3M=</body>
      <footer xsi:type="xsd:base64Binary" />
      <msisdn xsi:type="xsd:string">14155551212</msisdn>
      <uniqueid xsi:type="xsd:base64Binary" />
```

```
      <set_reply_path xsi:type="xsd:boolean">false</set_reply_path>
      <deliver_after xsi:type="xsd:date" />
      <price xsi:type="xsd:string">fteu</price>
      <campaign_id xsi:type="xsd:integer">xxx</campaign_id>
      <quios_sub xsi:type="xsd:string" />
      <quios_sub_id xsi:type="xsd:base64Binary" />
      <content_type xsi:type="xsd:string" />
    </namesp1:send_to_number>
  </soap:Body>
</soap:Envelope>
```

Listing 5-2 Example `send_to_numbers` request

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:namesp2="http://namespaces.soaplite.com/perl"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <namesp1:send_to_numbers xmlns:namesp1="http://soap.ewingz.com/eWingz/SOAP/
QC60">
      <username xsi:type="xsd:string">xxxxxxx</username>
      <password xsi:type="xsd:string">xxxxxxx</password>
      <testmode xsi:type="xsd:boolean">false</testmode>
      <notification xsi:type="xsd:string">handset</notification>
      <type xsi:type="xsd:string">GSM0338</type>
      <class xsi:type="xsd:integer">1</class>
      <udhi xsi:type="xsd:boolean">false</udhi>
      <originator xsi:type="xsd:base64Binary">xxxxx=</originator>
      <header xsi:type="xsd:base64Binary" />
      <body xsi:type="xsd:base64Binary">VmlzaXQgUXVpb3M=</body>
      <footer xsi:type="xsd:base64Binary" />
      <price xsi:type="xsd:string">99</price>
      <campaign_id xsi:type="xsd:integer">xxx</campaign_id>
      <numbers soapenc:arrayType="namesp2:HASH[2]" xsi:type="namesp2:ARRAY">
        <number xsi:type="namesp2:HASH">
          <msisdn xsi:type="xsd:string">14155551212</msisdn>
          <uniqueid xsi:type="xsd:base64Binary" />
          <set_reply_path xsi:type="xsd:boolean">false</set_reply_path>
          <deliver_after xsi:type="xsd:date" />
        </number>
        <number xsi:type="namesp2:HASH">
          <msisdn xsi:type="xsd:string">14155551212</msisdn>
          <uniqueid xsi:type="xsd:base64Binary" />
          <set_reply_path xsi:type="xsd:boolean">false</set_reply_path>
          <deliver_after xsi:type="xsd:date" />
        </number>
      </numbers>
      <quios_sub xsi:type="xsd:string" />
      <quios_sub_id xsi:type="xsd:base64Binary" />
      <content_type xsi:type="xsd:string" />
    </namesp1:send_to_numbers>
  </soap:Body>
</soap:Envelope>
```

Listing 5-3 Example `wap_push_si` request

```
  <?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <namesp1:wap_push_si xmlns:namesp1="http://soap.ewingz.com/eWingz/SOAP/QC60">
      <username xsi:type="xsd:string">xxxxxxx</username>
      <password xsi:type="xsd:string">xxxxxxx</password>
      <testmode xsi:type="xsd:boolean">false</testmode>
      <notification xsi:type="xsd:string">handset</notification>
      <originator xsi:type="xsd:base64Binary">xxxxx=</originator>
      <msisdn xsi:type="xsd:string">14155551212</msisdn>
      <uniqueid xsi:type="xsd:base64Binary" />
      <deliver_after xsi:type="xsd:date" />
      <price xsi:type="xsd:string">0</price>
      <campaign_id xsi:type="xsd:integer">xxx</campaign_id>
      <quios_sub xsi:type="xsd:string" />
      <quios_sub_id xsi:type="xsd:base64Binary" />
      <content_type xsi:type="xsd:string" />
      <href xsi:type="xsd:string">http://wap.yahoo.com</href>
      <action xsi:type="xsd:string" />
      <created xsi:type="xsd:string">2007-01-04T00:00:00Z</created>
      <si_expires xsi:type="xsd:string">2007-01-06T00:00:00Z</si_expires>
      <si_id xsi:type="xsd:string" />
      <text xsi:type="xsd:string">Visit yahoo</text>
    </namesp1:wap_push_si>
  </soap:Body>
</soap:Envelope>
```

Listing 5-4 Example `wap_push_sl` request

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <namesp1:wap_push_sl xmlns:namesp1="http://soap.ewingz.com/eWingz/SOAP/QC60">
      <username xsi:type="xsd:string">xxxxxxx</username>
      <password xsi:type="xsd:string">xxxxxxx</password>
      <testmode xsi:type="xsd:boolean">false</testmode>
      <notification xsi:type="xsd:string">handset</notification>
      <originator xsi:type="xsd:base64Binary">xxxxx=</originator>
      <msisdn xsi:type="xsd:string">14155551212</msisdn>
      <uniqueid xsi:type="xsd:base64Binary" />
      <deliver_after xsi:type="xsd:date" />
      <price xsi:type="xsd:string">fteu</price>
      <campaign_id xsi:type="xsd:integer">xxx</campaign_id>
      <quios_sub xsi:type="xsd:string" />
```

```
      <quios_sub_id xsi:type="xsd:base64Binary" />
      <content_type xsi:type="xsd:string" />
      <href xsi:type="xsd:string">http://wap.yahoo.com</href>
      <action xsi:type="xsd:string" />
    </namesp1:wap_push_sl>
  </soap:Body>
</soap:Envelope>
```

6 Message types and message classes

6.1 Regular Text messages (type GSM0338)

Characters intended for text display in the Message can be encoded by the ETSI GSM 03.38 default alphabet. A chart of the encodings and available characters are shown in Appendix A.

Different character sets are supported by different types of Handset hardware and mobile service carriers. Because Q-Caster cannot determine the specific character display capabilities for any particular Handset, text Messages are not guaranteed to display correctly. However, many Handsets and mobile service carriers, particularly those used outside the United States, are capable of transmitting and displaying ETSI GSM 03.38 characters. In most cases these characters transmit and display correctly. For the US, Quios will attempt to translate ETSI GSM 03.38 characters to the character set used by each of the major US carriers. Therefore, we recommend Providers to strictly adhere to the ETSI GSM 03.38 character set.

Note: Many characters in ETSI GSM 03.38 are similar to ASCII and Latin-1, which can give misleading testing results. Calling Applications should undergo thorough testing of unusual characters to ensure that the encoding is correct.

6.1.1 Line breaks in SMS messages

As mentioned previously, the display of characters on Handsets can vary between manufacturers and phone models. However, many phones will end a line and begin the next line in response to this ETSI GSM 03.38 character code: `0x0a` hexadecimal.

6.1.2 Long messages

Text messages of more than 140 bytes (160 characters in GSM0338) are split into multiple SMS transmissions of 140 bytes each. If the message must be split in a particular place (between words, for example), then the Calling Application must submit it as multiple messages, split in the proper place.

6.2 Double-byte Character Text messages (type UCS2)

Characters intended for text display in the Message can be encoded by the UCS2 character set. See the information on the Unicode Standard at http://www.unicode.org/ for more information.

Different subsets of UCS2 are supported by different types of Handset hardware and mobile service carriers. Because Q-Caster cannot determine the specific character display capabilities for any particular Handset, text Messages are not guaranteed to display correctly.

6.2.1 Long messages

Text messages of more than 140 bytes (70 UCS2 characters) are split into multiple SMS transmissions of 140 bytes each. If the message must be split in a particular place (between words, for example), then the Calling Application must submit it as multiple messages, split in the proper place.

6.3 8-bit binary messages (type Binary)

Q-Caster accepts messages in 8-bit binary format. It performs no validation or encoding; the Calling Application is responsible for proper validation and encoding.

The user data header and binary body are limited to 280 characters, with the space character bringing the total to 281. The SMS header is not included in this length limitation.

Each binary SMS is limited to a maximum of 140 encoded bytes and 281 total characters. However, the Calling Application can encode the `body` as an array of base64 items instead of a single base64 item. This single Message submission is delivered as multiple SMS messages. The SMSs are reassembled upon reaching the Handset.

Binary capabilities are dependent on the Handset's manufacturer and model.

Note: Although Quios cannot provide detailed support regarding binary content, online resources are available to assist in assembling binary content, particularly for Nokia Smart Messaging. Refer to the Nokia Smart Messaging Spec or the Nokia Smart Messaging FAQ for assistance with Smart Messaging formats.

Both of these documents are available, after registering, from the Nokia website http://www.forum.nokia.com/. For the Smart Messaging Spec, choose Technologies, then Messaging, then Documents. For the Smart Messaging FAQ, choose Technologies, then Messaging, then Smart Messaging.

Note: The Calling Application is responsible for setting the MCC/MCN octets of operator logos.

Table 6-1 Example values for binary logo submission

Parameter	Type	Value
`Username`	string	`myusername`
`Password`	string	`mypassword`
`Testmode`	boolean	`false`
`notification`	string	`handset`
`Type`	string	`Binary`
`Class`	integer	`1`
`Udhi`	boolean	`true`
`Originator`	base64Binary	
`Header`	base64Binary	
`Body`	array of base64Binary	`0B05041582000000030102013082F0100A00480E010 00000000000000000000007FFFFFFC000000003F821 F1043FC000007CF924F3249FBE00039FF924F3249FF DC01C7FF920F3241FFF3830FFF92191243FFF9C31FF F82493049FFFCC38FFF824F3049FFF8C3C7FF920F32 41FFF1C1F9FF921F3243FFC7803E0FFFFFFFFFF81C0 007C00FFFFFF80`
`Body`	array of base64Binary	`0B05041582000000030102023E000003FC0000003FC 000`
`Footer`	base64Binary	
`Msisdn`	string	
`Uniqueid`	base64Binary	

6.4 RTTTL messages (type RTTTL)

Instead of a text message for display on the Handset, the `body` can consist of an RTTTL ringtone for transmission to the Handset as a Smart Message. Q-Caster supports transmission of RTTTL. Lengthy ringtones are automatically split into multiple concatenated SMSs and need not be split before transmission to Q-Caster.

The Calling Application can set the RTTTL attributes for the message. The <name> attribute is required; behavior of RTTTL messages without a <name> attribute is unspecified. Other attributes are optional. For the attributes listed in Table 6-1, default values will be provided when the Calling Application omits values for these attributes.

Table 6-2: RTTTL attributes

Attribute name	Default value
duration	4
scale	6
beats	63
looping	0

The complete specification for RTTTL is in Appendix B. The values listed therein are acceptable; all other values cause indeterminate results.

RTTTL capabilities are dependent on the Handset's manufacturer and model; results on hardware that is not Smart Message-capable are undefined. Ringtone support is available only on certain hardware (such as Nokia models 3210, 3310, 6110, 6130, 6150, 6210, 6250, 7110, 8110i, 8210, 8810, 8850, 8890, 9110, 9110i, 9210). This information is subject to change.

Note: Certain types of errors in the RTTTL content might cause the message to be identified as a text message, not an RTTTL message. For example, using a hyphen "-" character in a song title is not valid in RTTTL, and will cause Q-Caster to identify and deliver the message as a series of SMS text messages. In addition, using an invalid tempo (such as tempo=3) will cause the message to fail identification as an RTTTL submission.

6.5 Message Class

6.5.1 Class 0: Flash

GSM0338 text messages can be sent as flash SMS messages (class 0 messages). The content of these messages appears on the Handset immediately, and is not stored to memory. Flash capabilities are dependent on the Handset's manufacturer and model; results on hardware that is not flash-capable are undefined.

6.5.2 Class 1: Messages delivered to memory

Text messages (both GSM0338 and UCS2) can be sent as SMS messages delivered to memory (class 1 messages). These are "ordinary" text messages. Their content doesn't appear on the Handset immediately, but is stored to memory. This is the recommended setting for message class, guaranteed to work across all carriers.

6.5.3 Class 2: Messages delivered to SIM

This feature is rarely used, but is included in the Q-Caster capabilities for completeness.

6.5.4 Class 3: Messages delivered to serial port

This feature is rarely used, but is included in the Q-Caster capabilities for completeness.

6.6 Message Originators

6.6.1 Dynamic Originators

Q-Caster's `originator` field allows the Provider to supply a customized originator for each SOAP Request. If a non-zero-length `originator` is supplied, then each SMS resulting from the `send_to_number` or `send_to_numbers` call will reflect that originator. In this way the originator can by dynamic; it can be set individually for each message.

Or, the SOAP Request can use a zero length `originator`, which causes Q-Caster to default to the `originator` text that was supplied with the Provider's account setup.

Note: the use of dynamic originators is not allowed in the US. All MT messages need to have an approved short code originator.

6.6.2 The originator field

The requirements for the originator field depend on the Provider's contractual agreement with Quios. Quios allows different types of originators. These specifications for originators apply to both the default originator field and to dynamic originator settings.

A standard originator must contain at least one non-numeric character, must be at least 2 and no more than 11 alphanumeric characters, and must match the following regular expression:

```
/^[-.+0-9A-Za-z]{2,11}$/
```

With prior contractual agreement, a Provider can use an International originator. With this type of originator, the Handset can call directly back to the message sender as represented in the originator. This type of originator must use a valid international format MSISDN in the originator field, optionally preceded by the plus "+" character. International originators must be at least 5 and no more than 15 digits. A leading plus "+" character forces the number to be interpreted as an international number.

With prior contractual agreement, a Provider can use a National originator (also called a Short Code). This type of originator must at least one and no more than 5 digits, and may not include a plus "+" character. A leading minus "-" character forces the number to be interpreted as a local number.

6.7 Scheduled delivery

Messages can be prepared and submitted to Q-Caster in advance of their intended delivery time to handsets. The `deliver_after` parameter reflects the date and time, in GMT, at which the message will leave the Q-Caster system for delivery by a downstream service provider.

Due to delays inherent in the nature of SMS, Q-Caster cannot guarantee an arrival time for any particular message; however, the message will not arrive at the handset prior to the date and time specified in `deliver_after`.

At any time before the messages leave the Q-Caster system, their delivery can be cancelled with the `cancel` method. Messages cannot be cancelled after leaving the Q-Caster system. If Q-Caster splits a large message into multiple SMS segments, then a cancel command might affect only some of those segments. Multiple SMS segments are delivered independently and are assembled upon reaching the handset. To ensure proper cancellation of large messages, issue the `cancel` command at least a few minutes before the `deliver_after` time occurs.

6.8 Content Type

This parameter is used on a per message basis and gives an indication of the content of that particular message (informational, alert, ringtone, etc.). The parameter must be specified FOR EVERY MESSAGE sent to AT&T's network (standard rate messages, FTEU messages, premium rate messages, etc.). If omitted, Quios will automatically assign the 'default' content_type for the campaign associated with the campaign_id. This parameter is only used by AT&T, but can be submitted for all messages and all carriers. The content_type will be ignored for all carriers, except AT&T.

Table 6-3 content_type Values

String	Content	Description
OTH	Other	Content which falls outside other defined types.
RGR	Ringtones	Ringtones
GMS	Games	Includes quizzes, sweepstakes, games and game downloads.
CHT	Chat	Chat, instant messaging and dating services.
INF	Info	Informational messages, double opt-in messages, premium billed 'thank you' messages.
MLT	Entertainment	Entertainment services (e.g. cinema, tickets, entertainment guides)
SPT	Sport	Sport related services
SCS	Visual Content	Logos, wallpapers, screensavers, animations.
VTE	Vote	Voting, polling, trivia, etc.
ALT	Alerts	Alerts

6.9 Quios_sub and quios_sub_id fields

Certain carriers such as AT&T require that all subscription management be under their direct control, enabling them to manage subscriptions with their End-Users without involving content providers. The quios_sub and quios_sub_id fields are used to manage the interaction between Providers and Carriers related to subscription management. For details on how to use these fields, please refer to the 'AT&T subscription management guide'.

7 Responses to messages

Each successful SOAP request receives a SOAP response; in Q-Caster, these responses are used to communicate to the Calling Application information about the message's progress through its delivery lifecycle. The initial SOAP response to `send_to_number`, `send_to_numbers`, `wap_push_si`, or `wap_push_sl` reports the initial status of the Request. Major error conditions result in SOAP faults, as described in Section 7.6.

The initial response indicates the result of initial parsing of the Request, including preliminary validity checks on the MSISDN and other contents of the Request. A `send_to_number` response is a `send_to_number_result` struct, which is described in Section 7.1.

A `send_to_numbers` response is an array. Each element of the array is a `send_to_number_result` struct, which is described in Section 7.1.

The original Request assigns one `uniqueid` to each Message, or allows Q-Caster to assign this `uniqueid`. However, if any Message is too large for a single SMS, then Q-Caster divides it into multiple SMSs ("segments") before delivery. The number of segments of a message is reported in the result.

Note: A successful response does **not** indicate successful delivery to the Handset. This preliminary status indicates only that Q-Caster has determined that the request is valid, and will send the request to its downstream providers. For additional information about the message's status, the Calling Application must call the `status` or `history` method, as described in Section 8.

Q-Caster responses always include numeric codes and associated text. The numeric codes, such as `disposition_code`, are the definitive result and should be used for the Calling Application's logic. The associated text, such as `disposition_text`, is provided solely for the convenience of the developer, and should not be referred to directly in the Calling Application.

7.1 The `send_to_number, wap_push_si, and wap_push_sl` result

The `send_to_number` result, `wap_push_si` result, or `wap_push_sl` result is used to convey information about the current or historical status of each SMS. See Table 7-1 for a list of the information returned in this struct. Following sections describe each of these keys.

Table 7-1 Values of `send_to_number, wap_push_si, and wap_push_sl` result

Key	Value Type	Meaning
`uniqueid`	base64 Binary	The `uniqueid` for this Message as provided by the Calling Application, or as generated by Q-Caster.
`segments`	integer	If the Message is too large for a single SMS, then Q-Caster divides it into multiple SMSs. This number indicates the total number of SMSs for this Message.
`response_code`	integer	A 3-digit code indicating the most recent action performed on the SMS. See 7-4 for more info.
`response_text`	string	A human-readable string that corresponds to the `response_code`. This string is subject to change at any time.
`disposition_code`	integer	A 1-digit code indicating the most recent status of the SMS. See

Key	Value Type	Meaning
		Table 7-3 for all legal values.
carrier_id	string	A numeric code, prefaced with P or C, indicating the carrier that delivered the SMS to the Handset. Requests with testmode=true return a placeholder carrier_id; the actual carrier_id will be returned if the Request is repeated with testmode=false. See 7-5 for more info.
campaign_id	integer	A numeric code indicating the campaign to which the submitted message belongs.
price	integer	Premium price associated with the message
disposition_text	string	A human-readable string that corresponds to the disposition_code. This string is subject to change at any time.

Listing 7-1 Example `send_to_number` result

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <namesp58:send_to_numberResponse xmlns:namesp58="http://soap.ewingz.com/eWi
ngz/SOAP/QC60">
      <send_to_number_result>
        <uniqueid xsi:type="xsd:base64Binary">xxxxxxxxxx=</uniqueid>
        <segments xsi:type="xsd:integer">1</segments>
        <response_code xsi:type="xsd:integer">110</response_code>
        <response_text xsi:type="xsd:string">message accepted</response_text>
        <disposition_code xsi:type="xsd:integer">2</disposition_code>
        <carrier_id xsi:type="xsd:string">C1022</carrier_id>
        <campaign_id xsi:type="xsd:integer">xxx</campaign_id>
        <price xsi:nil="true" xsi:type="xsd:integer" />
        <disposition_text xsi:type="xsd:string">processing</disposition_text>
      </send_to_number_result>
    </namesp58:send_to_numberResponse>
  </soap:Body>
</soap:Envelope>
```

Listing 7-2 Example `send_to_numbers` result

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <namesp5:send_to_numbersResponse xmlns:namesp5="http://soap.ewingz.com/eWin
gz/SOAP/QC60">
      <send_to_numbers_result soapenc:arrayType="xsd:anyType[2]"
                                                        xsi:type="soapenc:Array">
```

```
        <send_to_number_result>
          <uniqueid xsi:type="xsd:base64Binary">xxxxxxxxxx=</uniqueid>
          <segments xsi:type="xsd:integer">1</segments>
          <response_code xsi:type="xsd:integer">124</response_code>
          <response_text xsi:type="xsd:string">Response string</response_text>
          <disposition_code xsi:type="xsd:integer">2</disposition_code>
          <carrier_id xsi:type="xsd:string">C1022</carrier_id>
          <campaign_id xsi:type="xsd:integer">1</campaign_id>
          <price xsi:type="xsd:integer">-1</price>
          <disposition_text xsi:type="xsd:string">processing</disposition_text>
        </send_to_number_result>
        <send_to_number_result>
          <uniqueid xsi:type="xsd:base64Binary">xxxxxxxxxx=</uniqueid>
          <segments xsi:type="xsd:integer">1</segments>
          <response_code xsi:type="xsd:integer">124</response_code>
          <response_text xsi:type="xsd:string">Response string</response_text>
          <disposition_code xsi:type="xsd:integer">2</disposition_code>
          <carrier_id xsi:type="xsd:string">C1022</carrier_id>
          <campaign_id xsi:type="xsd:integer">xxx</campaign_id>
          <price xsi:type="xsd:integer">-1</price>
          <disposition_text xsi:type="xsd:string">processing</disposition_text>
        </send_to_number_result>
      </send_to_numbers_result>
    </namesp5:send_to_numbersResponse>
  </soap:Body>
</soap:Envelope>
```

Listing 7-3 Example `wap_push_si` result

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <namesp6:wap_push_siResponse xmlns:namesp6="http://soap.ewingz.com/eWingz/S
OAP/QC60">
      <wap_push_si_result>
        <uniqueid xsi:type="xsd:base64Binary">xxxxxxxxxx=</uniqueid>
        <segments xsi:type="xsd:integer">1</segments>
        <response_code xsi:type="xsd:integer">124</response_code>
        <response_text xsi:type="xsd:string">Response string</response_text>
        <disposition_code xsi:type="xsd:integer">2</disposition_code>
        <carrier_id xsi:type="xsd:string">C1022</carrier_id>
        <disposition_text xsi:type="xsd:string">processing</disposition_text>
        <campaign_id xsi:type="xsd:integer">1</campaign_id>
      </wap_push_si_result>
    </namesp6:wap_push_siResponse>
  </soap:Body>
</soap:Envelope>
```

Listing 7-4 Example `wap_push_sl` result

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <namesp6:wap_push_slResponse xmlns:namesp6="http://soap.ewingz.com/eWingz/S
OAP/QC60">
      <wap_push_sl_result>
        <uniqeid xsi:type="xsd:base64Binary">xxxxxxxxxx=</uniqeid>
        <segments xsi:type="xsd:integer">1</segments>
        <response_code xsi:type="xsd:integer">124</response_code>
        <response_text xsi:type="xsd:string">Response string</response_text>
        <disposition_code xsi:type="xsd:integer">2</disposition_code>
        <carrier_id xsi:type="xsd:string">C1022</carrier_id>
        <disposition_text xsi:type="xsd:string">processing</disposition_text>
        <campaign_id xsi:type="xsd:integer">1</campaign_id>
      </wap_push_sl_result>
    </namesp6:wap_push_slResponse>
  </soap:Body>
</soap:Envelope>
```

7.2 Notification levels

The extent of information available for a particular message depends on its progress through its delivery lifecycle, and on its notification level. The `notification` parameter is set in the `send_to_number(s)` request when the message is initially submitted to Q-Caster, **not** when the message status is requested. The notification level changes the information available to the Calling Application regarding the message's progress through its delivery lifecycle. For example, redelivery is attempted for 48 hours for every message that encounters an unavailable Handset, but final delivery confirmation is available only for those sent whose `notification` is `handset`. Without this notification level, information about the redelivery results is unavailable to the Calling Application (and to Quios).

According to the notification level, different types of information are available regarding the message's status. See Table 7-2 for an explanation of the notification levels. Subsequent tables will also refer to notification levels.

Table 7-2 Notification levels

Level	Meaning
`none`	Status of message lifecycle up to the point where it exits the Quios system for a downstream provider. Includes information on validity checks (for number length, country code, etc.), blocked numbers, and other preliminary results. Does not report final delivery to Handset.
`quios`	Status of entire message lifecycle for most messages. Includes final success/failure after first delivery attempt and immediate retries. Does not include status of delayed retries (as used when Handset is off or unreachable).

Level	Meaning
handset	Status of entire message lifecycle, including final delivery success/failure after immediate and delayed retries.

7.3 Message disposition: disposition_code and disposition_text

The message disposition indicates the progress of the message during the process of acceptance, testing, and transmission. A higher-numbered disposition state indicates that the message has progressed further in the transmission process. The message moves through the disposition states in order, but does not necessarily experience each state. See Table 7-3 for a listing of the values for disposition_code.

The last three dispositions, disposition_code 4, 5, and 6 (success, failure, and final), indicate that this message has reached the end of its transmission lifecycle, and no further action will be performed on this message.

Table 7-3 Corresponding disposition_code and disposition_text values

Code	Text	Meaning
1	Tested	Message was sent in testmode; tests were successful; if testmode were off then this message would be passed downstream.
2	Processing	Message is active in Quios system.
3	Waiting	Quios is waiting for status information from a downstream provider.
4	Success	Message was delivered to Handset.
5	Failure	Message will not be delivered to Handset.
6	Final	Message is finished in Quios system, but final status cannot be determined.

7.4 Message response: response_code and response_text

The message response indicates the results of actions performed on the message. See the document *Quios Response Codes* for a listing of the legal values of response_code. This document is available from Quios technical support or online at http://www.quios.com/docs/Quios_respcodes.pdf

In send_to_number response or send_to_numbers response, the response_code will be limited to responses to the preliminary validity testing that the message undergoes. Messages in testmode will also be limited to these responses. These responses are in the range of 100 to 199.

Note: The Calling Application must use only response_code in its logic; response_text values are for convenience only and are subject to change.

7.5 Carrier Information: `carrier_id`

Qcaster responses will contain the result of the dynamic reverse number-lookup performed by Quios on all incoming messages. The carrier information is returned in one of two formats:

Cxxxx -or-

Pxx

Quios will return the carrier information in the 'Pxx' format whenever the carrier is known to support short code based programs. This greatly simplifies the carrier management on the Provider's side, especially in North America, where each carrier has a significant amount of OCN number, each corresponding to a separate carrier_id. Quios has mapped all these id's into groups for carriers that support short codes. These group numbers are returned in the 'Pxx' format as per the following table:

Table 7-5 Premium Carrier Codes

ID	Premium Carrier
P1	AT&T Mobility (old Cingular)
P2	AT&T Mobility (old AT&T)
P3	Nextel
P4	Verizon Wireless
P5	Sprint PCS
P6	T-Mobile USA
P7	Alltel
P8	Cellular One
P9	US Cellular
P10	Dobson Cellular
P21	Boost Mobile

For carriers who don't support short codes, Quios will return the carrier information in the 'Cxxxx' format. For the most current list of carriers and ids, please execute the `carrier_list` or `coverage_map` methods as described in paragraph 11-3.

7.6 SOAP faults

Major errors in the message request result in SOAP faults. These faults indicate unrecoverable errors, and the entire SOAP request is rejected. These faults are generally not considered to be part of normal Calling Application operations, and should be

encountered rarely (if ever) in the Calling Application's production environment. In particular, Client.Authentication and Client.Input faults should never be encountered outside the new account setup and development activities.

A representative listing of SOAP faults is in Table 7-6; this listing is subject to change without notice. These faults can contain variable text, indicated by italics in the table.

Table 7-6: SOAP faults

Type	Text	Meaning
Server	General Failure - Untrapped Error	Error in Quios server; please send diagnostic information to support@quios.net. Include raw XML sent and XML received.
Client .Authentication	*username*/*password*/*IP* combination does not authenticate	The authentication information is not valid; check that the username, password, and IP address are authorized and correct.
Client.Input	`history` received unknown `uniqueid`	No Message was ever sent with this `uniqueid`; validate the `uniqueid` against the local database before submitting request.
Client.Input	`history` received wrong number of parameters: *count*, must be 3	The `history` request was malformed.
Client.Input	received empty message content, `header` + `body` + `footer` must be > 0	Message content was zero length; validate for empty content before submitting request.
Client.Input	received invalid `class`: *value*, must be 0, 1, 2 or 3	Value for `class` was not valid; validate for parameter values before submitting request.
Client.Input	received invalid notification: *value*, must be none, quios, or handset	Value for `notification` was not valid; validate for parameter values before submitting request.
Client.Input	received invalid `originator`: *value*	Value for `originator` was not valid; validate for parameter values before submitting request.
Client.Input	received invalid type: *value*, must be GSM0338, Binary, RTTTL, UCS2	Value for `type` was not valid; validate for parameter values before submitting request.
Client.Input	`send_to_number` received invalid msisdn, *msisdn*	Value for `misisdn` was not valid; validate for parameter values before submitting request.
Client.Input	`send_to_number` received too long `uniqueid`, *length* bytes, maximum 80	Value for `uniqueid` was not valid; validate for parameter values before submitting request.
Client.Input	`send_to_number` received wrong number of parameters:	The `send_to_number` request was malformed.

Type	Text	Meaning
	count, must be 20	
Client.Input	`send_to_numbers` received invalid msisdn, *msisdn*	Value for `misisdn` was not valid; validate for parameter values before submitting request.
Client.Input	`send_to_numbers` received non-array numbers parameter, numbers must be an array	The `send_to_numbers` request was malformed.
Client.Input	`send_to_numbers` received non-struct numbers element	The `send_to_numbers` request was malformed.
Client.Input	`send_to_numbers` received too long `uniqueid`, *length* bytes, maximum 80	Value for `uniqueid` was not valid; validate for parameter values before submitting request.
Client.Input	`send_to_numbers` received wrong number of parameters: *count,* must be 17	The `send_to_numbers` request was malformed.
Client.Input	`status` received unknown `uniqueid`	No Message was ever sent with this `uniqueid`; validate the `uniqueid` against the local database before submitting request.
Client.Input	`status` received wrong number of parameters: *count*, must be 3	The status request was malformed.
Client.Input	`quios_sub_id` is set and `quios_sub` is not set.	`quios_sub_id` is used in conjunction with the `quios_sub` field and cannot be used without setting the `quios_sub` field. See paragraph 6.9 for details.
Client.Input	`quios_sub` is START and `quios_sub_id` is set.	`quios_sub_id` field cannot be set when initiating a new subscription. See paragraph 6.9 for details.
Client.Input	`quios_sub` is not START and `quios_sub_id` is not set.	`quios_sub_id` must be set when modifying an existing subscription. See paragraph 6.9 for details.
Client.Input	Invalid `content_type`: Must be oth, rgr, gms, cht, inf, mlt, spt, scs, vte or alt.	Value for `content_type` was not valid; check paragraph 6.8 for details.

8 Checking message status

Q-Caster offers additional SOAP RPCs for retrieving information about how a Message moves through its delivery lifecycle. The extent of information available is determined by the Message's `notification` level, as explained in Section 7.2.

8.1 The `status` request and `history` request

To receive information on the current delivery status of a Message, the Calling Application submits a `status` request referring to the Message's `uniqueid`. This request will retrieve information about where the Message is in its lifecycle.

To receive information on the current and historical delivery status of a Message, the Calling Application submits a `history` request referring to the Message's `uniqueid`. This request will retrieve information about the Message's lifecycle, from the initial response to the current information.

For the types of data available via `status` request and `history` request, see Section 8.4.

Table 8-1 Parameters to `status` request and `history` request

Parameter	Type	Constraints	Meaning
`username`	string	must be valid username for the submitting IP address	Provider's username for authentication.
`password`	string	must be valid password for this username	A valid password for the username; used for authentication.
`uniqueid`	base64 Binary	must be valid `uniqueid` for this username	A unique string to identify the Message; multiple SMSs can have the same `uniqueid` if they were split automatically. Used to reference Messages for checking delivery status.

8.2 The `status` response

Each `status` request is answered with a `status` response, which is an array called `status_result`. This response indicates the current status of the Message in its delivery lifecycle. Although the `status` response reports the status of a single Message, it is nonetheless an array of `sms_status_query` structs, because the `status` request for information on a single `uniqueid` can refer to a number of SMSs if the original Message was split due to length. See Section 8.4 for information on `sms_status_query`.

Listing 8-1 Example of status response

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <namesp7:statusResponse xmlns:namesp7="http://soap.ewingz.com/eWingz/SOAP/Q
C60">
      <status_result soapenc:arrayType="xsd:anyType[1]" xsi:type="soapenc:Array">
        <sms_status_query>
          <response_code xsi:type="xsd:integer">328</response_code>
          <response_text xsi:type="xsd:string">message selected for campaigning
</response_text>
          <disposition_code xsi:type="xsd:integer">2</disposition_code>
          <disposition_text xsi:type="xsd:string">processing</disposition_text>
```

```
          <campaign_id xsi:type="xsd:integer">xxx</campaign_id>
          <price xsi:nil="true" xsi:type="xsd:integer" />
          <date xsi:type="xsd:date">2007-07-30</date>
          <time xsi:type="xsd:time">16:37:09Z</time>
        </sms_status_query>
      </status_result>
    </namesp7:statusResponse>
  </soap:Body>
</soap:Envelope>
```

8.3 The `history` response

Each `history` request is answered with a `history` response. This response contains all the current and historical information regarding the progress of the Message in its delivery lifecycle. The `history` response consists of an outer array, `history_result`. Each element of `history_result` is an `sms_history_result` array consisting of `sms_status_query` structs. See Section 8.4 for information on `sms_status_query`.

Listing 8-2 Example of history response

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <namesp7:historyResponse xmlns:namesp7="http://soap.ewingz.com/eWingz/SOAP/
QC60">
      <history_result soapenc:arrayType="soapenc:Array[1]" xsi:type="soapenc:Ar
ray">
        <sms_history_result soapenc:arrayType="xsd:anyType[N]" xsi:type="soapen
c:Array">
          <sms_status_query>
            <response_code xsi:type="xsd:integer">124</response_code>
            <response_text xsi:type="xsd:string">new campaign message enters sy
stem</response_text>
            <disposition_code xsi:type="xsd:integer">2</disposition_code>
            <disposition_text xsi:type="xsd:string">processing</disposition_tex
t>
            <campaign_id xsi:type="xsd:integer">xxx</campaign_id>
            <price xsi:nil="true" xsi:type="xsd:integer" />
            <date xsi:type="xsd:date">2007-07-30</date>
            <time xsi:type="xsd:time">16:37:08Z</time>
          </sms_status_query>
          <sms_status_query>
            <response_code xsi:type="xsd:integer">327</response_code>
            <response_text xsi:type="xsd:string">campaignable sms defunneled</r
esponse_text>
            <disposition_code xsi:type="xsd:integer">2</disposition_code>
            <disposition_text xsi:type="xsd:string">processing</disposition_text>
            <campaign_id xsi:type="xsd:integer">xxx</campaign_id>
            <price xsi:nil="true" xsi:type="xsd:integer" />
            <date xsi:type="xsd:date">2007-07-30</date>
            <time xsi:type="xsd:time">16:37:08Z</time>
```

```
          </sms_status_query>
          [...]
                    </sms_history_result>
      </history_result>
    </namesp7:historyResponse>
  </soap:Body>
</soap:Envelope>
```

8.4 The sms_status_query

The `sms_status_query` reports an individual state of an individual SMS. It is used in an array to report on all the SMSs that resulted from a Message submission. See Table 8-2 for the values returned by `sms_status_query`.

Table 8-2 Values of `sms_status_query`

Key	Value Type	Meaning
`response_code`	integer	An integer that specifies type of response
`response_text`	string	A human-readable string that corresponds to the `response_code`. This string is subject to change at any time.
`disposition_code`	integer	An integer that specifies type of disposition
`disposition_text`	string	A human-readable string that corresponds to the `disposition_code`. This string is subject to change at any time.
`campaign_id`	integer	A numeric code indicating the campaign to which the submitted message belongs.
`price`	integer	Premium price associated with the message
`date`	date	Date of relative sms_status_query
`time`	time	Timestamp of relative sms_status_query

9 Retrieving Mobile Originated (MO) Messages

Accounts that are properly configured for 2-way messaging can use Q-Caster to retrieve MO messages in their queue. Contact your Quios sales representative for information on 2-way messaging.

9.1 The `get_messages` request

To access messages in the 2-way queue, the Calling Application sends a `get_messages` request. The parameters of this RPC are listed in Table 9-1. A `get_messages` request will retrieve all available messages for this account, and remove them from the queue.

Table 9-1 Parameters to `get_messages` requests

Parameter	Type	Constraints	Meaning
`username`	string	must be valid username for the	Provider's username for authentication.

Parameter	Type	Constraints	Meaning
		submitting IP address	
password	string	must be valid password for this username	A valid password for the username; used for authentication.

9.2 The get_messages_result response

The response to the get_messages request is get_messages_result, an array of sms_get_entry elements. Each sms_get_entry represents a message that has been queued on the Q-Caster system awaiting this request. Each element of the array has the key/value pairs listed in Table 9-2.

Table 9-2 Key/value pairs of sms_get_entry elements

Key	Value Type	Meaning
class	integer	Indicates the SMS class of GSM0338 messages; 0 is a flash message, 1 is delivered to memory.
udhi	boolean	udhi=true shows that this message was sent with UDHI indicated. Q-Caster does not verify its UDHI compliance.
originator	base64	Contains the originator of the message.
msisdn	string	Contains the destination MSISDN (phone number) for this message.
body	base64	Contains the message body (message text).
carrier_id	string	A code indicating the carrier from whom we received the MO SMS.
date	date	Indicates the date that the message was received.
time	time	Indicates the time that the message was received.
uniqueid	base64	A unique string to identify the Message; multiple SMSs can have the same uniqueid if they were split automatically. Used to reference Messages for checking delivery status
campaign_id	Integer	ID identifying the campaign associated with the MO message received.
quios_sub	string	The action that was taken on a subscription. If NULL, the message is considered to be not subscription related
quios_sub_id	base64	A unique string that identifies the subscription, which corresponds to the uniqueid of the message that started the subscription
content_type	string	Specifies what type of content is sent with this text message. See 6.12 for details.

Listing 9-1 Example of get_messages_result response

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

```
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <namesp527:get_messagesResponse
xmlns:namesp527="http://soap.ewingz.com/eWingz/SOAP/QC60">
      <get_messages_result SOAP-ENC:arrayType="xsd:ur-type[N]" xsi:type="SOAP-
ENC:Array">
              <sms_get_entry>
                <class xsi:type="xsd:integer">0</class>
                <udhi xsi:type="xsd:boolean">1</udhi>
                <originator xsi:type="SOAP-ENC:base64">xxxxx=</originator>
                <msisdn xsi:type="xsd:string">14155551212</msisdn>
                <body xsi:type="SOAP-ENC:base64">xxxxxxxxxx=</body>
                <carrier_id xsi:type="xsd:string">xxxx</carrier_id>
                <date xsi:type="xsd:date">2003-05-17</date>
                <time xsi:type="xsd:time">15:57:43Z</time>
                <uniqueid xsi:type="SOAP-ENC:base64">xxxxxxxxxx=</uniqueid>
                <campaign_id xsi:type="xsd:integer">xxx</campaign_id>
                <quios_sub xsi:type="xsd:string">xxx</quios_sub>
                <quios_sub_id xsi:type="SOAP-ENC:base64">xxx=</quios_sub_id>
                <content_type xsi:type="xsd:string">xxx</content_type>
              </sms_get_entry>
              <sms_get_entry>
                <class xsi:type="xsd:integer">0</class>
                <udhi xsi:type="xsd:boolean">1</udhi>
                <originator xsi:type="SOAP-ENC:base64">xxxxx=</originator>
                <msisdn xsi:type="xsd:string">14155551212</msisdn>
                <body xsi:type="SOAP-ENC:base64">xxxxxxxxxx=</body>
                <carrier_id xsi:type="xsd:string">xxxx</carrier_id>
                <date xsi:type="xsd:date">2003-05-17</date>
                <time xsi:type="xsd:time">15:58:43Z</time>
                <uniqueid xsi:type="SOAP-ENC:base64">xxxxxxxxxx=</uniqueid>
                <campaign_id xsi:type="xsd:integer">xxx</campaign_id>
                <quios_sub xsi:type="xsd:string">xxx</quios_sub>
                <quios_sub_id xsi:type="SOAP-ENC:base64">xxx=</quios_sub_id>
                <content_type xsi:type="xsd:string">xxx</content_type>
              </sms_get_entry>
  [...]
      </get_messages_result>
    </namesp527:get_messagesResponse>
  </soap:Body>
</soap:Envelope>
```

10 Cancelling message delivery

[TO BE MIGRATED FROM QCASTER 4.0 – USE QC40 URL/URI/NAMESPACE FOR NOW]

Messages submitted for future delivery can be cancelled with the `cancel` request at any time before the resulting SMS or SMSs leave the Q-Caster system.

Large Messages are split by Q-Caster into multiple SMSs (“segments”). Each SMS is cancelled independently. To ensure that all segments of a Message are cancelled, submit the `cancel`

request sufficiently in advance of the `deliver_after` time that none of the segments has left the Q-Caster system.

Submitting a `cancel` request for a messages whose `deliver_after` parameter was missing or blank **might** successfully cancel the message if the message has not yet left the Q-Caster system. However, message cancellation is guaranteed **only** if the message has a scheduled delivery time, and if that delivery time has not yet been reached.

10.1 The `cancel` request

The Calling Application sends the `cancel` request to cancel the delivery of a Message that has not yet reached its `deliver_after` time. The Message is referenced by its `uniqueid` returned in response to the original Message submission. The parameters to the `cancel` request are shown in Table 10-1.

Table 10-1 Parameters to `cancel` requests

Parameter	Type	Constraints	Meaning
`username`	string	must be valid username for the submitting IP address	Provider's username for authentication.
`password`	string	must be valid password for this username	A valid password for the username; used for authentication.
`uniqueid`	base64Binary	must be valid `uniqueid` for this username	A unique string to identify the Message; multiple SMSs can have the same `uniqueid` if they were split automatically.

10.2 The `cancel` response

The response to the `cancel` request is `cancel` response. This response indicates the number of segments that the original Message generated, and the number of segments that was successfully cancelled. See Table 10-2 for the values returned by `cancel`.

Table 10-2 Values of `cancel` response

Key	Value Type	Meaning
`uniqueid`	base64	A unique string to identify the Message; multiple SMSs can have the same `uniqueid` if they were split automatically.
`segments`	integer	Indicates the number of SMS segments created from the original Message.
`cancelled`	integer	Indicates the number of SMS segments that were cancelled successfully.

Listing 10-1 Example of `cancel` response

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
```

```
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi=http://www.w3.org/1999/XMLSchema-instance
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <namesp1:cancelResponse
      xmlns:namesp1="http://soap.ewingz.com/eWingz/SOAP/QC40">
      <cancel_result>
        <uniqueid xsi:type="SOAP-ENC:base64">UUR2WnVqLVRrQkFBQUg4QkV2cw==</uniqueid>
        <segments xsi:type="xsd:integer">1</segments>
        <cancelled xsi:type="xsd:integer">1</cancelled>
      </cancel_result>
    </namesp1:cancelResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

11 Retrieving carrier and coverage information

[TO BE MIGRATED FROM QCASTER 4.0 – USE QC40 URL/URI/NAMESPACE FOR NOW]

Q-Caster provides methods to dynamically retrieve information that changes rapidly, such as the carrier associated with a particular Handset, the carriers currently reachable through the Q-Caster system, and the countries served by the current mix of available carriers.

11.1 The `number_lookup` request

The `number_lookup` request retrieves information about the current carrier for a particular Handset. The information reflects the carrier that will handle final delivery of the SMS, even if the Handset is roaming. The parameters to the `number_lookup` request are shown in Table 11-1.

The number lookup feature is a premium service offered by Quios. For information on enabling this service, contact your Quios sales representative.

Table 11-1 Parameters to the `number_lookup` request

Parameter	Type	Constraints	Meaning
`username`	string	must be valid username for the submitting IP address	Provider's username for authentication.
`password`	string	must be valid password for this username	A valid password for the username; used for authentication.
`msisdn`	string	minimum 7 digits, maximum 15 digits. International format. No spaces	Indicates the MSISDN (phone number) of the Handset.

Parameter	Type	Constraints	Meaning
		or alpha characters allowed. The + character is not allowed.	

11.2 The `number_lookup` response

Quios assigns a unique id to each carrier. The `number_lookup` response contains the name of the carrier currently providing service to the Handset.

Because the `number_lookup` response does not reference the `msisdn`, the Calling Application should wait for a response from each `number_lookup` request before submitting the next `number_lookup` request.

Table 11-2 Values of `number_lookup` response

Key	Value Type	Meaning
`id`	integer	Quios carrier id
`name`	string	Name of carrier associated with this id.

Listing 11-1 Example `number_lookup` response

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <namesp158:number_lookupResponse
xmlns:namesp158="http://soap.ewingz.com/eWingz/SOAP/QC40">
      <number_lookup_result>
                  <id xsi:type="xsd:integer">1022</id>
                  <name xsi:type="xsd:string">Cingular Wireless</name>
      </number_lookup_result>
    </namesp158:number_lookupResponse>
  </soap:Body>
</soap:Envelope>
```

11.3 The `carrier_list` request and `coverage_map` request

The carriers available to Quios are in constant flux, and the carriers available to any particular account are limited. The `carrier_list` request lists all the carriers and their capabilities; `coverage_map` returns the carriers currently available to the account. The parameters to these methods are shown in Table 11-3. It is assumed that the Calling Application keeps a local cache of the carrier list, then uses `coverage_map` more frequently to maintain fresh data regarding currently available carriers.

These methods can return large SOAP arrays. The Calling Application must be prepared to accept these large arrays.

Table 11-3 Parameters to `carrier_list` and `coverage_map` requests

Parameter	Type	Constraints	Meaning
`username`	string	must be valid username for the submitting IP address	Provider's username for authentication.
`password`	string	must be valid password for this username	A valid password for the username; used for authentication.

11.4 The `carrier_list` response

Each `carrier_list` request is answered with a `carrier_list` response, which is an array of `carrier`. See Section 11.6 for more information on `carrier`.

Listing 11-2 Example `carrier_list` response

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <namesp11:carrier_listResponse xmlns:namesp11="http://localhost/eWingz/SOAP/QC40">
      <carrier_list soapenc:arrayType="xsd:anyType[N]" xsi:type="soapenc:Array">
       <carrier>
                    <id xsi:type="xsd:integer">1</id>
                    <name xsi:type="xsd:string">CARRIER NAME</name>
                    <technology xsi:type="xsd:string">AMPS</technology>
                    <country xsi:type="xsd:string">CARRIER COUNTRY</country>
                     <ocn xsi:type="xsd:string">0</ocn>
                  </carrier>
                  [...]
      </carrier_list>
    </namesp11:carrier_listResponse>
  </soap:Body>
</soap:Envelope>
```

11.5 The coverage_map response

Each coverage_map request is answered with a coverage_map response, which is the name/id pair for each carrier currently available to this account.

Listing 11-3 Example coverage_map response

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <namesp9:coverage_mapResponse xmlns:namesp9="http://localhost/eWingz/SOAP/QC40">
      <coverage_map soapenc:arrayType="xsd:anyType[N]" xsi:type="soapenc:Array">
                    <carrier>
                      <id xsi:type="xsd:integer">1</id>
                      <name xsi:type="xsd:string">CARRIER NAME</name>
                    </carrier>
                    <carrier>
                      <id xsi:type="xsd:integer">2</id>
                       <name xsi:type="xsd:string">CARRIER NAME</name>
                    </carrier>
                    [...]
      </coverage_map>
    </namesp9:coverage_mapResponse>
  </soap:Body>
</soap:Envelope>
```

11.6 The carrier request

The carrier request reports the information on a carrier and its capabilities. It is used in an array to show all the carriers available to Quios or to a particular account. See Table 11-4 for the values returned by carrier.

Table 11-4 Values of carrier

Key	Value Type	Meaning
id	integer	Quios carrier id
name	string	Name of carrier associated with this id.
technology	string	GSM, TDMA etc
country	string	The physical location served by this carrier.
OCN	string	Operating Company Number; defined for NANP US carriers only.

12 The Quios Permission Management System (QPMS)

The QPMS service handles all carrier-specific requirements regarding Consumer opt-in, opt-out, spending caps, etc. Its main goal is to shield any carrier-specific complexity from the Provider. This means that the Provider will never have to take any specific action depending on the carrier who delivered the MO message. All actions, specific to one or more carriers, are implemented by the QPMS service and are totally transparent to the Provider.

The QPMS service is based on the notion of campaigns, services, and actions.

a. Campaign: is a carrier-approved program, using a common short code as identifier, to which Consumers can sign up. Multiple Campaigns can be used on the same short code. Each Campaign will be identified by the unique keyword used when signing up for the Campaign. Campaigns running on the same short code share common keywords such as STOP and HELP.

b. Service: is a specific set of Actions undertaken as part of a Campaign. The result can be to opt-in a Consumer, or opt him out, or provide help, or other types of information.

c. Action: is a single execution element. A Service consists of multiple Actions that are strung together and executed in sequential order.

QPMS interfaces with the Provider using standard MO and MT messages (send_to_number(s), and get_messages methods). These methods now include an additional parameter, campaign_id. Quios will set up a unique campaign_id for each new campaign the Provider uses on a particular short code. Provider needs to submit the campaign_id with every MT message. Quios will use context sensitive business logic to associate a campaign with every MO received, and passes the campaign_id to the Provider.

QPMS implements following features: (i) campaign look-up; (ii) subscriber management; (iii) opt-in management; (iv) opt-out management; (v) context sensitive help; (vi) spending cap management; (vii) error management.

12.1 Campaign lookup

For every MT message, Provider will specify the campaign_id. Example: if Provider runs a daily horoscope campaign (campaign id 1) and a joke-of-the-day campaign (campaign id 2) on the same short code, Provider will specify campaign_id=1 for every horoscope MT delivered to the Quios gateway, and campaign_id=2 for every joke MT delivered to the Quios gateway.

For every MO message, Quios will identify the campaign associated with the MO message and uses following information to to determine the right campaign: (i) short code; (ii) subscriber info; (iii) body of the message; (iv) history of the subscriber. The campaign id will be passed on to the Provider as part of the get_message_response.

12.2 Subscriber management

Quios will maintain a list of active subscribers on a per campaign basis. This allows Provider to run multiple campaigns on the same short code with distinct subscriber databases. It will also allow Provider to provide campaign specific information to the Consumer even when that

Consumer requests information using ambiguous keywords such as HELP, or STATUS. Because QPMS is context sensitive, it will provide information related to a specific campaign, even when the keyword is not specific.

12.3 Opt-in management

QPMS will handle carrier-specific opt-in requirements totally transparently to Provider. Opt-in requirements vary by carrier in two ways: (i) content of the opt-in messages; (ii) message flow variations.

a. Content of the opt-in message. Some carriers have specific content requirements. Example: Tmobile requires all opt-in messages to include the words 'standard rates apply'. Nextel/Boost on the other hand require all opt-in messages to include the words 'other rates apply'. QPMS handles these requirements without Provider involvement.

b. Message flow variations. Depending on the type of campaign and billing, some carriers may require double opt-in, where as other carriers are happy with a single opt-in method. QPMS handles both opt-in methods withour Provider involvement.

Once a Consumer has completed the opt-in process, Provider will receive an MO message with the original keyword provided by the Consumer when he sent his first MO message.

The text of all messages is fully customizable.

12.4 Opt-out management

QPMS will handle carrier-specific opt-out requirements totally transparently to Provider. Opt-out requirements vary by carrier in two ways: (i) content of the opt-out message; (ii) encoding of the opt-out response message.

a. Content of the opt-out message. Some carriers have specific content requirements. Example: Tmobile requires all opt-out responses to specify that the Consumer will no longer receive any messages or charges.

b. Encoding of opt-out response. Cingular Orange requires all opt-out responses to be 'free to the end user', where as other carriers require responses to be 'standard rated'.

Once a Consumer has completed the opt-out process, Provider will receive an MO message with the original keyword provided by the Consumer.

The text of all messages is fully customizable.

12.5 Context-sensitive Help

QPMS handles all HELP requests by implementing the mandatory keyword 'HELP'. The response provided by QPMS will be relative to the campaign to which the consumer subscribed, or will be a generic HELP if the consumer has not subscribed to any campaign. The encoding of the HELP response will be implemented in accordance with carrier specific requirements ('Free to end user' on Cingular Orange, 'standard rated' on all other carriers) and the body of the message will have the right content as per the specific carrier requirements.

The text of all messages is fully customizable.

12.6 Spending Cap Management (Premium SMS only)

Carriers require that consumer spending is monitored and capped on a monthly basis. When the consumer reached the monthly cap, all premium SMS messages should be blocked until the end of the month, unless the consumer agrees to an increase in the cap (Tmobile only).

QPMS handles these requirements transparently to the Provider. CAP requirements vary by carriers in three ways: (i) the actual dollar amount a consumer can spend per month; (ii) the ability for a consumer to increase the cap by triggering an additional double opt-in message; (iii) the actual text of the message.

The text of all messages is fully customizable.

12.7 Error Handling

QPMS uses a sophisticated error handling mechanism to deal with consumer errors. What if a consumer enters the wrong keyword ? What if they reply ‘YESS’ instead of ‘YES’ to confirm opt-in? QPMS will return context sensitive error information to the consumer to assist the consumer throughout the opt-in process.

The text of all messages is fully customizable.

Appendix A: Available GSM/UCS2 Characters and Their Encodings

http://www.unicode.org/Public/MAPPINGS/ETSI/GSM0338.TXT provides information about GSM0338 characters and how they map to Unicode characters.

For information on UCS2 characters, see The Unicode Standard at http://www.unicode.org

Characters used by ETSI GSM 03.38 default alphabet

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	@	£	$	¥	è	é	ù	ì	ò	ç	LF	Ø	ø	C	Å	å
10	Δ	_	Φ	Γ	Λ	Ω	Π	Ψ	Σ	Θ	Ξ	es	Æ	æ	ß	É
20	spc	!	"	#	¤	%	&	'	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	¡	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	Ä	Ö	Ñ	Ü	§
60	¿	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	ä	ö	ñ	ü	à

Appendix B: RTTTL Specification

The values listed here are acceptable; any other values can cause indeterminate results.

```
<ringing-tones-text-transfer-language> :=
<name> <sep> [<defaults>] <sep> <note-command>+

<name> := <char>+ ; maximum name length 10 characters

<sep> := ":"

<defaults> :=
<def-note-duration> := 'd'
<def-note-scale>    := 'o'
<def-beats>         := 'b'
<def-style>         := 's'
<def-looping>       := 'l'

If not specified, defaults are

4  = duration
6  = scale
63 = beats-per-minute

Valid in tone section: o, b, s

 <note-command> :=
[<duration>] <note> [<scale>] [<special-duration>] <delimiter>

<duration> :=
"1"  Full 1/1 note
"2"  1/2 note
"4"  1/4 note
"8"  1/8 note
"16"1/16 note
"32" 1/32 note

<note> :=
"P"  pause
"C"
"C#"
"D"
"D#"
"E"
"F"
"F#"
"G"
"G#"
"A"
"A#"
"B"
"H"
```

```
<scale> :=
"4"  Note A is 440Hz
"5"  Note A is 880Hz
"6"  Note A is 1.76 kHz
"7"  Note A is 3.52 kHz

<special-duration> :=
"."  Dotted note
";"  Double dotted note
"&"  2/3 length

<delimiter> := ","

Acceptable values for B are:
25|28|31|35|40|45|50|56|63|70|80|90|100|112|125|140|160|180|
200|225|250|285|320|355|400|450|500|656|635|715|800|900

Acceptable values for Volume are: 1 through 15, inclusive
```

Appendix C: Known Issues/Bugs

Date	Description of issue
2007-08-01	The methods of Paragraph 10 and 11 are currently not supported by Qcaster6.0. Use the Qcaster4.0 namespace URI reference instead.
2007-08-01	The Quios messaging platform is processing incoming parameters as arrays (instead of hashes). As a result, it's critically important that Providers submit parameters in the correct order.
2007-08-02	The originator parameter in the wap_push_si and wap_push_sl methods is currently ignored and replaced with the short code associated with the campaign, as per the submitted campaign_id

Appendix D: Document change log

Date	Section	Description of change
2007-08-13	4, 7.6	Updated URI/URL references and SOAP Faults table
2007-08-01	All	Updated xml code examples
2007-07-30	all	new document for QC6.0