January 2015

XML API Specification V2.5.1

Clickatell
Unlock Possibilities

Contents

1. Overview

This technical document is intended for developers who wish to use the Clickatell FTP API for sending messages and describes the various programming methods and commands used by developers when using this API.

XML stands for 'Extensible Markup Language' and allows developers and designers to create their own tags. It provides a basic syntax that allows the creation of customer markup language that can be used across different platforms without having to pass through layers of conversion. The XML API can be used through an Internet browser.

To use this API, you need to register at (http://www.clickatell.com/register/?product=1). When you sign up for an XML account you will be given a username, password and api_id: keep these at hand. Once you have registered and been activated you will receive 10 free credits with which to test our service. Messages sent with these credits contain a pre-populated Clickatell message. You can test the API using these credits, and purchase credits to start sending your own, customized messages.

It is recommended that you understand routing profiles before reading this document. Information is available at at https://www.clickatell.com/resources/product-help/developers-central/routing-profile-guide/.

There are several different ways of gaining access to the gateway:

- SMTP - enabling a server or client generated email to be delivered as an SMS
- HTTP / HTTPS - submitting either a POST or GET to the API server
- FTP – uploading a text file to our FTP Server
- XML – posting to our gateway using XML over HTTP/S
- COM Object – for Windows based development
- SOAP – submit SOAP packets over HTTP/S
- SMPP – customers requiring a high throughput binary socket connection

Testing the Clickatell Gateway

Clickatell offers a test number range which will assist in reducing testing costs. Messages sent to any number on this prefix will only be charged 1/3 of a credit. When testing the Clickatell gateway you can use the number 279991xxxxx (for South Africa) or 1999xxxxxxx (for the U.S.) where "xxxxx" represents any numeric string. The status of your messages will be returned.

In order to reduce testing costs, Clickatell offers a test number range. Messages sent to any number on this prefix will only be charged 1/3 of a credit. Use the number 279991xxxxx (for South Africa) or 1999xxxxxxx (for the U.S.) where "xxxxx" represents any numeric string. Message statuses will be returned.

We will cover the **XML** method in this document. Additional documentation is available for the other methods. Sample code is provided on the site.

2. Introduction

Quite often, the easiest way for two applications to speak to each other is via XML. The XML is submitted via an HTTP post. Appendix C provides the full XML API dtd.

Note: It is important that the ENTIRE document is read before contacting support. You must use character references for Greek and other extended characters.

3. Getting started

Step 1 - register for a Clickatell account
If you do not already have a Developers' Central account, you need to register for one. If you already have a Clickatell Central account, proceed to Step 2 for instructions on how to edit an API connection on your account.

- Go to https://www.clickatell.com/clickatell-products/online-products/sms-gateway-developers-central/ and click on the 'Try Developers' Central Now' button.
- Select the Developers' Central and the Account type you would like to use.
- Enter your personal information to complete the registration form
- Accept Terms & Conditions
- Click the 'Create my Account' button - an email containing your login details will be sent to the email address you have provided.

Step 2 – Login to your account
When you have logged in you will be on the Clickatell Central landing page. You will receive 10 free credits which you can use to test the Clickatell Gateway. Please note that for security reasons these 10 credits contain pre-set Clickatell content.

An HTTP API will be added to your account for you. This will allow you to start testing the Clickatell Gateway immediately. You can purchase credits when you are ready to start sending personalized messages.

Step 3 – Adding an XML API to your account
To add an XML API to your account, select **APIs** from the main menu and then select **Setup a new API** from the submenu. Click the Add XML API button on the Setup API page that opens. You can then complete all the required details to configure your API.

After successfully adding a connection, a confirmation message will be displayed with a unique API ID and information on how to get started.

The getting started section displays the API connection parameters and authentication details. These details are required when connecting to the Clickatell gateway to send a message.

Note: For more information on managing your API connections within your Clickatell account see our API guide at http://www.clickatell.com/help-support/developer-apis/clickatell-api/

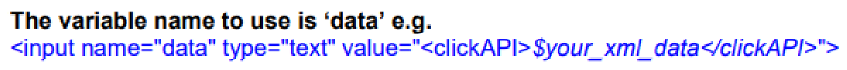
4. Basic XML structure

Root Tag: clickAPI

This tag must surround all your other API calls with each post. Remember calls are case-sensitive:
<click API>   your data... </clickAPI>

5. Submitting the XML to the gateway

Submit your case-sensitive XML as an HTTP form post to the following URL:
http://api.clickatell.com/xml/xml


The variable name to use is 'data' e.g.

```
<input name="data" type="text" value="<clickAPI>$your_xml_data</clickAPI>">
```

There is a test form to post XML data into at
http://api.clickatell.com/xml/tst.html

6. Basic commands (tags)

The following sections describe each of the tags used in the XML specification in more detail. The response tag that is returned is the name of the submit tag with "Resp" appended. For example, **auth** becomes **authResp.**

6.1 Authentication and session IDs

In order to deliver a message, the system needs to authenticate the request as coming from a valid source. We use a number of parameters to achieve this:

- **api_id:** This is issued to you when you register for the API product. A single Clickatell account may have multiple api_ids associated with it.
- **user:** This is the username of your account.
- **password:** The current password you have set on your account.

Additionally, we can enforce an IP lockdown, allowing only requests sent from IP addresses that you have specified under the API product preferences. Please ensure that after testing, you remove all unnecessary IP addresses in your preferences, to tighten up on security.

You can have multiple sessions open. However, the session ID will expire after fifteen minutes of inactivity. You will then have to re-authenticate to receive a new session ID. Alternatively you can ping every 10 minutes or so, to ensure that the current session ID is kept live.

This session ID must be used with all future commands to the API, unless you authenticate each time within the command itself.

Name	auth	
Parameters:	api_id	Required
	user	Required
	password	Required
	sequence_no	[Optional]
Name	authResponse	
ResponseVals:	session_id	
	or	
	fault	
	sequence_no	[If set]

6.2 Ping

This command prevents the session ID from expiring in periods of inactivity. The session ID is set to expire after 15 minutes of inactivity. You may have multiple concurrent sessions using the same session ID.

Name	ping	
Parameters:	session_id	Required
	sequence_no	[Optional]
Name	pingResponse	
ResponseVals:	ok	
	or	
	fault	
	sequence_no	[If set]

6.3 Send a message

To facilitate sending an SMS with a single command, we have included the ability to post **api_id**, **user** and **password** parameters in *sendmsg*. Using a session ID is preferred to authenticating each time.

Name	sendMsg	
Parameters:	session_id	Required
	to	Required
	text	Required
	callback	[Optional]
	cliMsgId	[Optional]
	concat	[Optional]
	deliv_ack	[Optional]
	deliv_time	[Optional]
	from	[Optional]
	msg_type	[Optional]
	udh	[Optional]
	unicode	[Optional]
	validity	[Optional]
	req_feat	[Optional]
	max_credits	[Optional]
	queue	[Optional]
	escalate	[Optional]
	sequence_no	[Optional]
Name	**sendMsgResp**	
ResponseVals:	apiMsgId	
	or	
	fault	
	sequence_no	[If set]

6.4 Query a message

This tag returns the status of a message. You can query the status with either the **apimsgid** or **climsgid**. The API message ID (**apimsgid**) is the message ID returned by the gateway when a message has been successfully submitted. If you specified your own unique client message ID (**climsgid**) on submission, you may query the message status using this value. You may also authenticate with **api_id**, **user** and **password**.

See Appendix B for status codes.

Name	queryMsg	
Parameters:	session_id	Required
	apiMsgId	Required
	or	
	cliMsgId	
	sequence_no	[Optional]
Name	queryMsgResp	
ResponseVals:	apiMsgId,status	
	or	
	fault	
	sequence_no	[If set]

Note: Clickatell can also post message status updates to your application via means of a Callback URL. This is the recommended method to obtain message status updates as your application is not required to continually poll the Clickatell gateway. Detailed information can be found in the "Callback URL" section under "Message Parameters".

Message statuses reports can be viewed online within your Central account. These reports can also be exported in CSV or Excel format.

7. Message parameters (tags)

7.1 Table of parameters

There are a variety of messaging and SMS features supported by the gateway, which can be activated by including a number of additional parameters. These parameters include those in the table below.

Name	Parameter name	Short description	Default value	Restricted values
API product ID	api_id	The value for this mandatory parameter can be found logging in online and going to **APIs -> Manage APIs**		
Username	user	The username you specified.		
Password	password	Your Developers' Central account password.		
Destination address	to	The number of the handset to which the message must be delivered. The number should be in international number format.		No '00' prefix or leading "+" symbol should be used.
Text	text	The text content of the message. Note that some characters take up two characters because of GSM encoding standards		Go to http://forums.clickatell.com/clickatell search for 'Why do some characters take two spaces?'
Source address	from	The source/sender address that the message will appear to come from also known as "Sender ID". These must be registered within your online account and approved by us before they may be used. MO numbers rented from us do not require approval.	gateway assigned number	A valid international format number between 1 and 16 characters long, or an 11-character alphanumeric string.
Enable callback	callback	Enables you to receive message delivery statuses via an HTTP, SOAP or XML callback which is posted to a URL of yours using the **GET** or **POST** method. This is done every time a message status is updated.	0	0,1,2,3,4,5,6,7 Read detailed description of parameter.
Delivery time	deliv_time	Delays delivery of SMS to mobile device in minutes relative to the time at which the SMS was received by our gateway. This should be greater than 10 minutes for best effect. Smaller time frames may be delivered too soon.		The upper limit is 7 days, or 10080 minutes.

Concatenation	concat	Specifies the maximum number of message parts available for the message.	1	1, 2, 3
Maximum credits	max_credits	Overrides the maximum charge specified online in “profiles”. It works within the bounds of the profiles. In other words, a profile must exist for the maximum credit that you set.	As per profiles	0.8,1,1.5,2,2.5,3
Required features	req_feat	Allows you to set the features which must be included when a message is sent. If the route does not support the features which you set as ‘required’ the message will fail. **Note:** The use of this parameter could increase the cost per message if a more expensive gateway is used.		Read detailed description of parameter.
Delivery queue	queue	Delivers the message through one of three queues assigned to each client account. Messages in the highest priority queue will be delivered first.	3	1, 2,3 1 is highest priority.
Gateway escalation	escalate	Prompts an escalation to an alternative route, if messages are queued on the least-cost route.	0	0 - off 1 - Escalate immediately to an alternative route if messages are queued on the least-cost route.
Mobile originated	mo	This is only applicable to clients that have subscribed to a two-way messaging service. We route via a pre-defined carrier to enable the ability for a reply to be received back.	0	0 – Off. We use our normal routing rules. 1 – Enable Reply.
Client message ID	cliMsgId	Client message ID defined by user for message tracking.		Up to 32 alphanumeric characters. No spaces.
Unicode message	unicode	Two-digit language code. Convert your text to Unicode [UCS-2 encoding]. See http://www.Unicode.org/.	0	0 – No Unicode 1 – Send as Unicode.

Message type	msg_type	Message types are associated with a structure that defines the fields of the message, e.g., logos and ringtones. See Message Types for more information.	SMS_TEXT	
User data header	udh	Informs the mobile handset of the type of data and data length of the user data part of an SMS message. The UDH header is used in conjunction with Binary content to define message types. See 8-bit messaging for more information.		Set UDH data manually.
Data	data	The data content of a message, if the UDH component is set manually.		
Validity period	validity	The validity period in minutes relative to the time at which the SMS was received by our gateway. The message will not be delivered if it is still queued on our gateway after this time.	1440 minutes (24 hours)	Set value in X minutes from 1 – 1440 minutes.

7.2 Message parameters in detail

7.2.1 Destination address <to>

SMS messages need to be sent in the standard international format, with country code followed by number. No leading zero to the number and no special characters such as "+" or spaces must be used. For example, a number in the UK being 07901231234 should be changed to 447901231234.

If the optional API setting titled *'Replace the leading zero with correct country code'* is enabled for the API in your Developers' Central account, any mobile numbers starting with zero will have the zero stripped and replaced with the international dialing code.

Text

This is the default parameter that is used to add message content. A single text message can contain up to 160 characters or 140 bytes.

7.2.2 Source address <from>

The source address (**from**), also known as the sender ID, can be either a valid international format number between 1 and 16 characters long, or an 11-character alphanumeric string. These must be registered within your online account and approved by us before they may be used. MO numbers rented from us do not require approval.

Note that characters such as spaces, punctuation, Unicode, and other special characters may not always be supported to all destinations and could interfere with your delivery. We suggest that you refrain from using such characters on the source address. The use of an alphanumeric source address with 8-bit messaging may cause message failure. This service is not guaranteed across all mobile networks and may interfere with delivery to certain handsets.

Note: To ensure that this feature is supported for MO numbers when your message is delivered, the required features (**req_feat**) parameter for this feature must be set.

7.2.3 Delivery acknowledgement <deliv_ack>

In order to determine whether an SMS has been received by a handset or not, we request delivery acknowledgement for every message we send. The ability to receive reliable delivery acknowledgements varies between mobile networks. Please test to a specific mobile network first, before assuming that you will receive handset acknowledgments for messages that are delivered.

If a GSM handset is ‘absent’, e.g., switched off or out of coverage, the SMS will be delivered according to a retry cycle once the handset is back in coverage. A delivery receipt will only be returned if and when the retry is delivered. If the validity period or retry cycle (typically 24 hours) is exceeded, the SMS will fail and show ‘Error Delivering Message’ or status 8.

Delivery acknowledgements can be monitored via the callback system or online reports.

7.2.4 Callback System <callback>

Final or intermediary statuses are passed back by the API depending on the **callback** value set in the original post. This is done by means of:

- HTTP GET
- HTTP POST
- XML GET
- XML POST
- SOAP GET
- SOAP POST

The variables returned are **api id, apiMsgId, cliMsgId, to, timestamp, from, status** and **charge**.

Validation of Callback URL

The URL entered in your Clickatell central account to receive 'SMS Status notifications' is validated to check if a callback can be completed. The URL must begin with either *http://* (non-encrypted) or *https://* (encrypted). If the callback URL is invalid, a message is displayed indicating an Invalid URL.

Callback retry interval

A retry mechanism allowing eight retries is activated if a status update is not delivered.

For Example:

1. 2 minutes after the original attempt
2. 4 minutes after last retry
3. 8 minutes after last retry
4. 16 minutes after last retry
5. 32 minutes after last retry
6. 64 minutes after last retry
7. 128 minutes after last retry
8. 3 days after last retry (max retries reached)

Optional Callback username and password

An optional "username" and "password" can be set in the preferences section of your API product. This username and passwords are not the same as your Clickatell username and password but is a setting of your choice to add additional security.

Callback value	Message status types returned	Message status code returned
0	No message status returned.	
1	Returns only intermediate statuses.	002, 003, 011
2	Returns only final statuses of a message.	004, 005, 006, 007, 008, 010, 012
3	Returns both intermediate and final statuses of a message.	All except 001

Examples

- **HTTP**

Sample callback to your callback URL using an HTTP get:

https://www.yoururl.com/script.asp?api_id=12345&apiMsgId=996f364775e24b8432f45d77da8eca47&cliMsgId=abc123&timestamp=1218007814&to=279995631564&from=27833001171&status=003&charge=0.300000

- **- XML**

The following data is sent in XML MT callbacks in a parameter called 'data':

```
<?xml version="1.0"?>
<callback>
 <apiMsgId>996411ad91fa211e7d17bc873aa4a41d</apiMsgId>
 <cliMsgId></cliMsgId>
 <timestamp>1218008129</timestamp>
 <to>279995631564</to>
 <from>27833001171</from>
 <charge>0.300000</charge>
 <status>004</status>
</callback>
```

Sample callback to your callback URL using an **XML GET**:

```
https://www.yoururl.com/script.php?data=<?xml
version="1.0"?><callback><apiMsgId>996411ad91fa211e7d17bc873aa4a41d</apiMsgId><cliMsgId></cliMsg
Id><timestamp>1218008129</timestamp><to>279995631564</to><from>27833001171</from><charge>0.30
0000</charge><status>004</status></callback>
```

- **SOAP**

With the SOAP callback method, a SOAP packet will be sent with a parameter called 'data'. Below is an example packet that will be sent to you via GET or POST.

Example of a SOAP packet that will be sent to you via **GET** or **POST**:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="mt_callback">
 <SOAP-ENV:Body>
```

```
   <tns:mt_callback xmlns:tns="mt_callback">
      <api_id xsi:type="xsd:int">1234</api_id>
      <apimsgid xsi:type="xsd:string">2e838df2ee3ea418272ae05aaf84ce5d</apimsgid>
      <climsgid xsi:type="xsd:string">abc123</climsgid>
      <to xsi:type="xsd:string">27999123456</to>
      <from xsi:type="xsd:string">27999000224</from>
      <timestamp xsi:type="xsd:int">1213690834</timestamp>
      <status xsi:type="xsd:int">003</status>
      <charge xsi:type="xsd:float">0.300000</charge>
    </tns:mt_callback>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

This is an example callback URL that will be sent to your application:

```
http://www.yoursite.com/your_url.php?data="<?xml version="1.0" encoding="ISO-8859-1"?><SOAP-
ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="mt_callback"><SOAP-ENV:Body>
<tns:mt_callback xmlns:tns="mt_callback"><api_id xsi:type="xsd:int">1234</api_id>
<apimsgid xsi:type="xsd:string">2e838df2ee3ea418272ae05aaf84ce5d</apimsgid><climsgid
xsi:type="xsd:string">abc123</climsgid><to xsi:type="xsd:string">27999123456</to>
<from xsi:type="xsd:string">27999000224</from><timestamp
xsi:type="xsd:int">1213690834</timestamp><status xsi:type="xsd:int">003</status>
<charge xsi:type="xsd:float">0.300000</charge></tns:mt_callback></SOAP-ENV:Body></SOAP-
ENV:Envelope>"
```

7.2.5 Delivery time <deliv_time>

The delivery of an SMS message may be delayed by setting an amount of time in **minutes** relative to the time at which it was received by our gateway. We will store the message until the required time frame has elapsed. The maximum delay time is 10080 minutes or 7 days. The actual delivery time of scheduled messages can always be handled up to 5 minutes too early.

When sending batches of messages, the delivery time should be set in the *startbatch* command. This will ensure that all messages are delivered X minutes after being posted to the Gateway.

7.2.6 Concatenation <concat>

If this value is set to 1, 2 or 3 the message will span across 1, 2 or 3 SMS messages where applicable. One text SMS will be sent for every 160 characters or 140 bytes. If a message is concatenated, it reduces the number of characters contained in each message by 3. With 8-bit concatenated messages, each SMS can support up to 140 bytes including the UDH headers.

Please also see http://support.clickatell.com/faq.php?mode=view_entry&kbid=121&kbcat=26 for information on characters that require two-character places.

Please be aware that a single Unicode SMS can only contain a maximum of 70 characters. If a Unicode message is concatenated, it reduces the number of characters contained in each message part by 7.

Values set are:

Value	Status
1	Default - No concatenation: only 1 message.
2	Concatenate a maximum of 2 messages.
3	Concatenate a maximum of 3 messages.
N	Concatenate a maximum of N messages. (Delivery is dependent on mobile and gateway. A maximum of 3 is recommended. The maximum number of messages that can be concatenated is 35)

7.2.7 Maximum credits <max_credits>

This parameter overrides the maximum charge associated with message delivery, as set by the profiles selected within your client account after logging in online. This parameter can be used to limit the cost of a message to a particular value and is bound by the maximum credit value specified in your profiles.

A valid API message ID can still be returned for messages that are not delivered as a result of the maximum credits value set. These messages will have a status of routing error (009).

The credit value in this parameter can be set to any amount of credits. To set your delivery profile, go to **Manage account -> Manage Routing Profiles.**

7.2.8 Required features (req_feat)

This parameter specifies the features that must be present in order for message delivery to occur. If all features are not present, the message will not be delivered. This prevents SMS messages arriving at a destination via the least-cost gateway, without certain features. This would, for instance, prevent the dropping of a sender ID.

This means that we will not route messages through a gateway that cannot support the required features you have set. For certain message types, we always set the required feature bitmask where relevant. These are FEAT_8BIT, FEAT_UDH, FEAT_UCS2 and FEAT_CONCAT.

This parameter is set using a combined decimal number to refer to the additional required features.

E.g.: 32 + 512 = 544 – Numeric sender ID and Flash SMS both required.
The value you would set to ensure that Flash and numeric sender ID are both supported, would therefore be ***544.***
To ensure that delivery acknowledgment and alphanumeric IDs are supported you would use the value ***8240*** (16 + 32 + 8192).

Hex value	Decimal	Feature	Description
0x0001	1	FEAT_TEXT	Text – set by default.
0x0002	2	FEAT_8BIT	8-bit messaging – set by default.
0x0004	4	FEAT_UDH	UDH (Binary) - set by default.
0x0008	8	FEAT_UCS2	UCS2 / Unicode – set by default.
0x0010	16	FEAT_ALPHA	Alpha source address (from parameter).
0x0020	32	FEAT_NUMER	Numeric source address (from parameter).
0x0200	512	FEAT_FLASH	Flash messaging.
0x2000	8192	FEAT_DELIVACK	Delivery acknowledgments.
0x4000	16384	FEAT_CONCAT	Concatenation – set by default.

7.2.9 Delivery queue <queue>

Setting this parameter will assign the message to one of three queues assigned to each user account. This sets the priority of a message sent to us, relative to other messages sent from the same user account. Messages in queue number 1, will always be delivered before messages in queue number 2 and 3, while messages in the 3rd queue, will have the lowest priority (relative to queues 1 and 2).

This is useful when delivering, for example, a single high priority message while you have a large batch going through that same account. The large batch will be queued through queue number 3 (default), and urgent alerts (sent through queue 1), will be delivered ahead of those messages in the batch (queue 3), regardless of when they are actually sent to us.

Values set are:

Value	Status
1	Use first / primary user queue (highest priority).
2	Use second user queue.
3	Use third user queue (lowest priority) - Default status.

7.2.10 Gateway escalation <escalate>

By default, the message router will select the lowest cost route (matching features and reliability) that is available for a given destination.

This parameter ensures that, should a message be delayed due to gateway congestion or some other reason on the initial gateway selected by our router, then alternative routes that match the required features will be sought. This is done by moving through the available gateways in order of increasing cost, up to the maximum charge set by the user either using the parameter that defines the maximum credits or based on the profiles selected.

When urgent and high priority messages are sent, they should be posted with escalate set to 1 (on), combined with a high maximum credit value to ensure that the greatest number of gateways are available.

Values set are:

Value	Status
0	Off – Default value.
1	On - Escalate immediately to an alternative route if the messages are queued on the least-cost route.

7.2.11 Mobile originated <mo>

This parameter is only used when a message is sent to a handset and a reply is expected.

PLEASE NOTE: This parameter is only valid for clients that have signed up and paid for our two-way messaging service. An alternative to our least-cost gateway may be used. This could result in a higher cost per message. Please email Clickatell support for pricing or view online.

When sending a normal MT message to a handset and you expect a reply to your registered MO number, please set the **MO** parameter to "1".

Values to set are:

Value	Status
0	Off - Default status. We use the normal routing feature.
1	Enables reply ability. We route via a pre-defined carrier to enable the ability to reply.

It is important that the user specifies the correct **from** parameter together with this parameter. If no **from** parameter is specified, we will use a default originator number as set by Clickatell. You will NOT receive these replies.

If you specify the originator (the purchased mo number), then we will route the message such that it can be replied to by the recipient. This reply will be sent to you.

7.2.12 Client message ID <cliMsgid>

This parameter is set by the user to enable internal message tracking. It allows the user to set their own tracking ID for each message. Once set for a given message, this may be used in place of the Clickatell issued API message ID (apimsgid) for querying message.

A client message ID (climsgid) may be any combination of alphanumeric characters excluding spaces. A maximum of 32 characters may be used.

Client message IDs may be used with the *querymsg* command.

7.2.13 Unicode <unicode>

If this value is set to 1, the text field must contain two-byte Unicode. Each SMS can handle a maximum of 70 characters. Each Unicode character must be hex encoded. More information is available at http://www.Unicode.org/.

Note: When using the batch send facility for delivering Unicode messages, it is not possible to substitute variables into the message content. This is only possible with Germanic characters.

Values set are:

Value	Status
0	Off – Default status
1	On - Delivers the text as two-byte Unicode.

We provide a converter to convert text to Unicode within your client account online. Go to "Converters" from within your account online.

7.2.14 Message type <msg_type>

A wide variety of messages can be sent through our gateway. We have pre-defined a number of SMS message-types in the API, so that you do not have to set the UDH (user data header) manually. You may optionally set the UDH rather than using one of the message types set below.

For non-Nokia message types (EMS, etc.), please generate your own UDH and data according to the manufacturer's specifications of the message type you wish to send.

This parameter need not be included if the SMS is a standard text message.

Values set are:

Value	Description
SMS_TEXT	This is the default message type. It is optional to specify this parameter.
SMS_FLASH	To send an SMS that displays immediately upon arrival at the phone.
SMS_NOKIA_OLOGO	Send an operator logo to a Nokia handset.
SMS_NOKIA_GLOGO	Send a group logo to a Nokia handset
SMS_NOKIA_PICTURE	Send a picture message to certain Nokia handsets.
SMS_NOKIA_RINGTONE	Send a ringtone to a Nokia handset
SMS_NOKIA_RTTL	Send an RTTTL format ringtone to Nokia handsets.
SMS_NOKIA_CLEAN	Remove operator logo from a Nokia handset.
SMS_NOKIA_VCARD	Send a business card to a Nokia handset.
SMS_NOKIA_VCAL	Send an event calendar to a Nokia handset.

7.2.15 Validity period <validity>

A message may be given a time frame for which it is valid. After this period the message will expire. This parameter takes an amount of time in **minutes** relative to the time at which the message was received by our gateway. If the message is queued on our gateway for a period exceeding the validity period set, then a routing error of 115 will be returned. The default validity period is 1440 minutes (24 hours).

Note: The validity period is not passed on to the upstream gateway.

8. Additional commands (tags)

Please note that where commands require a **session_id** for authentication or **api_id**, **username** and **password** can be used instead.

8.1 Delete/Stop message

This enables you to stop the delivery of a particular message. This command can only stop messages which may be queued within our router, and not messages which have already been delivered to a SMSC. This command is therefore only really useful for messages with deferred delivery times.

Name	delMsg	
Parameters:	session_id	Required
	apiMsgId	Required
	or	
	cliMsgId	
	sequence_no	[Optional]
Name	delMsgResp	
ResponseVals:	apiMsgId	
	or	
	fault	
	sequence_no	[If set]

8.2 Query balance

This will return the number of credits available on this particular account. The account balance is returned as a floating point value.

Name	getBalance	
Desc:	This enables you to determine your account balance	
Parameters:	session_id	Required
	sequence_no	[Optional]
Name	getBalanceResp	
ResponseVals:	ok	

	or	
	fault	
	sequence_no	[If set]

8.3 Coverage query

This command enables users to check our coverage of a network or number, without sending a message to that number. Authentication is required for this API call. This call should NOT be used before sending each message.

A response tag of ok indicates it is covered, while a fault tag indicates that the destination is not covered. Contact support to enquire about obtaining coverage for this destination.

Name	routeCoverage	
Parameters:	msisdn	Required
	session_id	Required
	sequence_no	[Optional]
Name	routeCoverageResp	
ResponseVals:	ok	
	charge	
	or	
	fault	
	sequence_no	[If set]

8.4 MMS Push

When an MMS message is sent to a phone, the mobile device receives an MMS notification message via SMS. When this MMS notification message is received by the mobile device, the mobile device automatically initiates a WAP gateway connection to download the content of the MMS message, from a URL specified in the SMS notification message. This command enables users to send an MMS notification message. Authentication is required for this API call.

MMS documentation (WAP-209-MMSEncapsulation-20020105-a.pdf, Version 05-Jan-2002) can be found at http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html.

Parameter	Description	Example	Default value	Restricted value	Required
mms_subjec t	Subject	My+message			yes
mms_class	Class	80		80 (Personal) 81 (Advertisement) 82 (Informational) 83 (Auto)	yes
mms_expire	How long before the MMS expires	3000		Time in seconds	yes
mms_from	From text	John			yes
mms_url	URL with the MMS content. The URL must be URL encoded.	http://www.mywebsite.com/example.mms			yes

Name	ind_push	
Parameters:	session_id	Required
	mms_subject	Required
	mms_class	[Optional]
	mms_expire	[Optional]
	mms_from	[Optional]
	mms_url	[Optional]
Name	sendMsgResp	
ResponseVals:	apiMsgId	
	or	
	fault	
	sequence_no	[If set]

8.5 WAP push service indication

WAP Push Service Indication (SI) is a WAP address embedded within the header of a specially formatted SMS. This is displayed as an alert message to the user and gives the user the option of connecting directly to a particular URL via the handsets WAP browser (if supported). This command enables you to send a WAP Push Service Indication.

Please note: Incorrect date formats for si_created and si_expires **may lead to handsets discarding messages with delivery receipts**.

WAP documentation (WAP-167-ServiceInd-20010731-a.pdf, Version 31-July-2001) can be found at http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html.

Parameter	Description	Example	Default value	Restricted values	Required
si_id	Unique ID for each message				yes
si_url	The URL that is used to access the service or content. The URL must be URL encoded.	http://www.65mydomain.com?picture=6566			yes
si_text	Notification text. Provides a means to specify additional information.	Here is your picture.			yes
si_created	A date in UTC (Coordinated Universal Time) format. Used to specify the date and time associated with the creation or last modification of the content indicated by the URL, which may differ from the date and time when the SI was created.	2008-01-01T19:30:41Z			no

si_expires	Expiry date in UTC format. This allows you to specify a time after which the SI will automatically be deleted from the handset. If not specified it will never expire.	2008-12-12T19:30:40Z			no
si_action	A string specifying the action to be taken when the SI is received.		signal-medium,	signal-none, signal-low, signal-medium, signal-high, delete.	no

Name	si_push	
Parameters:	session_id	Required
	si_id	Required
	si_url	[Optional]
	si_text	[Optional]
	si_created	[Optional]
	si_expires	[Optional]
	si_action	[Optional]
Name	sendMsgResp	
ResponseVals:	apiMsgId	
	or	
	fault	
	sequence_no	[If set]

8.6 Get message charge query

This command enables the user to query both the status and charge of a delivered message in a single API call. You can query the status with either the apimsgid or climsgid.

Authentication is required for this API call and will only work for messages less than 15 days old. Clickatell can also post the message charge to your application via means of a Callback URL (this is the preferred method). Detailed information can be found in the “Callback URL” section under “Message parameters”.

Name	getMsgCharge	
Parameters:	session_id	Required
	apiMsgId	Required
	sequence_no	[Optional]
Name	getMsgChargeResp	
ResponseVals:	apiMsgId	
	status	
	charge	
	or	
	fault	
	sequence_no	[If set]

8.7 Token (voucher) pay

This command allows you to spend a voucher that has been generated, within your client account, after logging in online. This is very useful for topping up sub-users accounts with credits. You would generate a session ID for the sub-user account, into which you wish to add the credits. You may also use the standard login details. The voucher number is currently a 16-digit numeric value.

Name	tokenPay	
Parameters:	session_id	Required
	token	Required
	sequence_no	[Optional]
Name	tokenPayResp	
ResponseVals:	<ok>605</ok>	If payment was successful
	or	
	<fault>ERR: 606, Invalid Voucher</fault>	
	or	

	<fault>607, Expired Token</fault>	
	or	
	<fault>609, Token Generation Error<fault>	

9. Batch messaging

This facility enables one to do high volume delivery and server-side message merging. It offers the end-user the ability to define all elements common to a batch, and then send only the parameters that change on a message-by-message basis.

One initially defines a batch using the *startbatch* command, which will return a unique batch ID. You then use either *senditem* or *quicksend* with the batch ID, depending on whether the message needs to be personalised. See SMS examples below.

Hi #field1#, your doctor's appointment is at #field2# tomorrow, could become:
Hi Fred, your doctor's appointment is at 10:30 tomorrow.
Hi Jane, your doctor's appointment is at 14:00 tomorrow.

9.1 Start batch

Once you have issued this command, you will be returned a batch ID that is to be used when sending multiple batch items. Included functionality also allows for message merging where you can substitute fields that you have defined in your template. The field names are called *field1* though to *fieldN.*

This command can take all the parameters of *sendmsg*, with the addition of a template, and the exception of both the destination address and the text fields. It must be used before either the *senditem* or *quicksend* command.

Name	startBatch	
Parameters:	session_id	Required
	template	Required
	callback	[Optional]
	cliMsgId	[Optional]
	concat	[Optional]
	deliv_ack	[Optional]
	deliv_time	[Optional]

	escalate	[Optional]
	from	[Optional]
	max_credits	[Optional]
	msg_type	[Optional]
	queue	[Optional]
	req_feat	[Optional]
	udh	[Optional]
	unicode	[Optional]
	validity	[Optional]
	sequence_no	[Optional]
Name	startBatchResp	
ResponseVals:	batch_id	
	or	
	fault	
	sequence_no	[If set]

9.2 Sending messages to existing batch

Send a message to a batch passing the destination mobile number and optional field replacement values. The fields 1-N that you defined in the *startbatch* command are used to optionally personalize the message.

Name	sendItem	
Parameters:	session_id	Required
	batch_id	Required
	to	Required
	fields	[Optional]
	sequence_no	[Optional]

Set the field values as per the template set up in *startbatch* command tag. This tag can only be used within *sendItem* command tag

Name	fields	
Parameters:	field1	Required
	field2	[Optional]
	field3	[Optional]
	field4	[Optional]
	field5	[Optional]
	field6	[Optional]
	field7	[Optional]
	field8	[Optional]
	field9	[Optional]
	field10	[Optional]
Name	sendItemResp	
ResponseVals:	apiMsgId	
	or	
	fault	
	sequence_no	[If set]

9.3 Quick send to batch

Where one has the requirement to send the same message to multiple recipients, you can use the *quicksend* command. This command offers low overhead and maximum throughput. It is essentially a reference to a predefined template and a string of destination addresses.

Name	sendItem	
Parameters:	session_id	Required
	batch_id	Required
	to	Required
	sequence_no	[Optional]
Name	quickSendResp	
ResponseVals:	apiMsgId	
	or	

	fault	
	sequence_no	[If set]

9.4 End batch

This command ends a batch and is not required (following a batch send). Batches will expire automatically after 24 hours.

Name	endBatch	
Parameters:	session_id	Required
	batch_id	Required
	sequence_no	[Optional]
Name	endBatchResp	
ResponseVals:	ok	
	or	
	fault	
	sequence_no	[If set]

10. 8-BIT messaging

Through the XML interface, one is also able to send 8-bit messages. These are most often used for ringtones and logos, but one can also send vCards, vCalendar appointments and EMS messages. When sending 8-bit messages, you need to set the user data header (UDH) of the SMS as well as sending the data. If you are comfortable with the creation of your own UDH, we also enable you to set it directly using the udh parameter. To simplify the process, we have provided a number of pre-defined message types (see the **msg_type** parameter).

With the standard ***text*** parameter, line breaks are automatically inserted. The parameter **data** is thus used for 8-bit messaging.

Example:
api_id:1234
user:xxxxxxxxx
password:xxxxxxxxxxx
to:xxxxxxxxxxxxxxxx

msg_type:SMS_NOKIA_RINGTONE
data:024A3A5585E195B198040042D9049741A69761781B6176156174288B525D85E0A26C24C49A61762
8 930BB125E055856049865885D200

11. Message examples

Here are some example URLs that demonstrate how to use the API. All values in these examples should be replaced by your own values.

11.1 Simple examples

sendmsg command including authentication and sender ID:

```
<clickAPI>
        <sendMsg>
                <api id>1</api id>
                <user>demo</user>
                <password>demo</password>
                <to>123456567890123</to>
                <text>Initial text message</text>
                <from>me</from>
        </sendMsg>
</clickAPI>
```

Initial authentication:

```
<clickAPI>
        <auth>
        <api_id>1</api_id>
                <user>demo</user>
                <password>demo</password>
        </auth>
</clickAPI>
```

All further commands will use a session ID generated using auth command above:

sendmsg command:

```
<clickAPI>
        <sendMsg>
                <session id>d9ef3ea7834f8a14618232d36375021a</session id>
                <to>123456567890123</to>
                <text>Initial text message</text>
```

```
                <from>me</from>
        </sendMsg>
</clickAPI>
```

Flash SMS:

```
<clickAPI>
        <sendMsg>
                <session id>d9ef3ea7834f8a14618232d36375021a</session id>
                <to>123456567890123</to>
                <msg type>SMS FLASH</msg type>
                <text>flash text message</text>
                <from>me</from>
        </sendMsg>
</clickAPI>
```

sendmsg with callback request:

```
<clickAPI>
        <sendMsg>
                <session_id>d9ef3ea7834f8a14618232d36375021a</session_id>
                <to>123456567890123</to>
                <text>Initial text message</text>
                <from>me</from>
               <callback>3</callback>
        </sendMsg>
</clickAPI>
```

Account balance:

```
<clickAPI>
        <getBalance>
                <session_id>d9ef3ea7834f8a14618232d36375021a</session_id>
        </getBalance>
</clickAPI>
```

Query message status:

```
<clickAPI>
        <queryMsg>
                <session_id>d9ef3ea7834f8a14618232d36375021a</session_id>
                <apiMsgId>4889e40291643afeb5a7c4cce7811abb</apiMsgId>
        </queryMsg>
</clickAPI>
```

Monitoring the connection (keeping the connection alive):

```
<clickAPI>
        <ping>
                <session_id>d9ef3ea7834f8a14618232d36375021a</session_id>
        </ping>
</clickAPI>
```

11.2 Batch SMS examples

11.2.1 Sending a personalized messages to multiple recipients

To start the batch:

```
<clickAPI>
        <startBatch>
                <session id>d9ef3ea7834f8a14618232d36375021a</session id>
                <template>Hi #field1# this is a personalised message</template>
        </startBatch>
</clickAPI>
```

To end the batch:

```
<clickAPI>
        <endBatch>
             <api_id>12345</api_id>
```

```
             <user>myUserName</user>
             <password>myPassword</password>
             <batch id>9b93a87b09d07e294c3f21131be400a0</batch id>
        </endBatch>
</clickAPI>
```

11.2.2 Sending two personalized messages

```
<clickAPI>
        <sendItem>
                <session id>d9ef3ea7834f8a14618232d36375021a</session id>
                <batch_id>f677d2fbb858a79aad0556dc71dd4383</batch_id>
                <to>1234567890</to>
                <fields>
                       <field1>David</field1>
                </fields>
        </sendItem>
        <sendItem>
                <session id>d9ef3ea7834f8a14618232d36375021a</session id>
                <batch id> f677d2fbb858a79aad0556dc71dd4383</batch id>
                <to>2345678901</to>
                <fields>
                       <field1>John</field1>
                </fields>
        </sendItem>
</clickAPI>
```

11.2.3 Sending multiple SMS using batches

```
<clickAPI>
        <quickSend>
                <session id>d9ef3ea7834f8a14618232d36375021a</session id>
                <batch_id>f677d2fbb858a79aad0556dc71dd4383</batch_id>
                <to>1234567890,2345678901,3456789012,4567890123</to>
        </quickSend>
</clickAPI>
```

11.3 8-bit SMS examples

Note: You cannot set an alphanumeric Sender ID (**from** parameter) when sending 8-bit messages.

11.3.1 Sending a ringtone

Sending a ringtone using the **msg_type** parameter:

```
<clickAPI>
        <sendMsg>
                <session id>d9ef3ea7834f8a14618232d36375021a</session id>
                <to>1234567890</to>
                <msg_type>SMS_NOKIA_RINGTONE</msg_type>
                <text>024A3A5585E195B198040042D9049741A69761781B6176156174288B525D85E0
A26C24C49A617628930BB125E055856049865885D200</text>
        </sendMsg>
</clickAPI>
```

Sending same ringtone with the **udh** parameter:

```
<clickAPI>
        <sendMsg>
                <session id>d9ef3ea7834f8a14618232d36375021a</session id>
                <to>1234567890</to>
                <udh>06050415810000</udh>
                <data>024A3A5585E195B198040042D9049741A69761781B6176156174288B525D85E0
A26C24C49A617628930BB125E055856049865885D200</data>
        </sendMsg>
</clickAPI>
```

11.3.2 Sending an operator logo

```
<clickAPI>
        <sendMsg>
                <session_id>d9ef3ea7834f8a14618232d36375021a</session_id>
                <to>1234567890</to>
                <msg_type>SMS_NOKIA_OLOGO</msg_type>
                <text>00480e010FC0000000000000003FF000000000000000
70380F9B006000001B601818DB006000C01BCF0C3058006000C01BDF8C301B3E66F9EF9BDF8C301B626C8CD8DBDF8C301B607
87CDFDBCF0C305B6078CCD81B601818DB626C8CD8DB70380F9B3E66FCEF9B3FF00000000000000000FC000000000000000000
00000000000000</text>
        </sendMsg>
</clickAPI>
```

11.3.3 Removing an operator logo

```
<clickAPI>
        <sendMsg>
                <session id>d9ef3ea7834f8a14618232d36375021a</session id>
                <to>1234567890</to>
                <msg type>SMS NOKIA CLEAN</msg type>
                <text>00</text>

        </sendMsg>
</clickAPI>
```

11.3.4 Sending a VCARD

```
<clickAPI>
        <sendMsg>
                <session_id>d9ef3ea7834f8a14618232d36375021a</session_id>
                <to>1234567890</to>
                <msg type> SMS NOKIA VCARD</msg type>
                <text>BEGIN%3AVCARD%0D%0AVERSION%3A2.1%0D%0AN%3ABloggs%3BJoe%0D
%0ATEL%3BPREF%3A%2B1234567890%0D%0AEND%3AVCARD%0D%0A</text>
        </sendMsg>
</clickAPI>
```

11.3.5 Sending a VCAL

```
<clickAPI>
        <sendMsg>
                <session_id>d9ef3ea7834f8a14618232d36375021a</session_id>
                <to>1234567890</to>
                <msg_type> SMS_NOKIA_VCAL</msg_type>
                <text>BEGIN%3AVCALENDAR%0D%0AVERSION%3A1.0%0D%0ABEGIN%3AV
TODO%0D%0ACATEGORIES%3AMISCELLANEOUS%0D%0ASUMMARY%3AMeet+buyers+at+Mario's%0D
%0ADTSTART%3A20030301T133000%0D%0AEND%3AVTODO%0D%0AEND%3AVCALENDAR%0D%0A
</text>
        </sendMsg>
</clickAPI>
```

11.4 Multi- Post MT

Submitting multiple API calls in one packet.

```
<clickAPI>
    <sendMsg>
        <session_id>d9ef3ea7834f8a14618232d36375021a<</session_id>
        <to>279992423223</to>
       <callback>1</callback>
        <text>This is a message.</text>
    </sendMsg>
    <sendMsg>
        <session_id>d9ef3ea7834f8a14618232d36375021a<</session_id>
       <callback>3</callback>
        <to>27999234234</to>
      <text>This is another message</text>
    </sendMsg>
```

```
</clickAPI>



Response:

<?xml version="1.0"?>
<clickAPI>
        <sendMsgResp>
               <apiMsgId>d36328fa5012a9ce151b797500598c9b</apiMsgId>
               <sequence no></sequence no>
        </sendMsgResp>
        <sendMsgResp>
               <apiMsgId>afdf2067566d48e9763c206bf56d0099</apiMsgId>
               <sequence_no></sequence_no>
        </sendMsgResp>
</clickAPI>
```

12. Appendix A: Error codes

The following list of error messages are generated by the Clickatell gateway during a validation phase before we accept the message. These error messages are sent back to your application. There will be no message charge if these errors are generated when sending a message. Data regarding messages that do not pass initial validation will not be included in your Clickatell Central reports.

Number	Description	Detail
001	Authentication failed	Authentication details are incorrect.
002	Unknown username or password.	Authorization error, unknown username or incorrect password.
003	Session ID expired	The session ID has expired after a pre-set time of inactivity.
005	Missing session ID	Missing session ID attribute in request.
007	IP Lockdown violation	You have locked down the API instance to a specific IP address and then sent from an IP address different to the one you set.
101	Invalid or missing parameters	One or more required parameters are missing or invalid.
102	Invalid user data header	The format of the user data header is incorrect.
103	Unknown API message ID	The API message ID is unknown. Log in to your API account to check the ID or create a new one.
104	Unknown client message ID	The client ID message that you are querying does not exist.
105	Invalid destination address	The destination address you are attempting to send to is invalid.
106	Invalid source address	The sender address that is specified is incorrect.
107	Empty message	The message has no content.
108	Invalid or missing API ID	The API message ID is either incorrect or has not been included in the API call.
109	Missing message ID	This can be either a client message ID or API message ID. For example, when using the stop message command.
113	Maximum message parts exceeded	The text message component of the message is greater than the permitted 160 characters (70 Unicode characters). Select **concat** equal to 1,2,3-N to overcome this by splitting the message across multiple messages.
114	Cannot route message	This implies that the gateway is not currently routing messages to this network prefix. Please email support@clickatell.com with the mobile number in question.
115	Message expired	Message has expired before we were able to deliver it to the upstream gateway. No charge applies.
116	Invalid Unicode data	The format of the Unicode data entered is incorrect.
120	Invalid delivery time	The format of the delivery time entered is incorrect.
121	Destination mobile number blocked	This number is not allowed to receive messages from us and has been put on our block list.

122	Destination mobile opted out	The user has opted out and is no longer subscribed to your service.
123	Invalid Sender ID	A sender ID needs to be registered and approved before it can be successfully used in message sending.
128	Number delisted	This error may be returned when a number has been delisted.
130	Maximum MT limit exceeded until <UNIX TIME STAMP>	This error is returned when an account has exceeded the maximum number of MT messages which can be sent daily or monthly. You can send messages again on the date indicated by the UNIX TIMESTAMP.
201	Invalid batch ID	The batch ID that you have entered for batch messaging is not valid.
202	No batch template	The batch template has not been defined for the batch command
301	No credit left	Insufficient credits
302	Max allowed credit	You have exceeded the maximum credit amount that you have set for messaging
606	Invalid Token	Invalid or no token was specified.
607	Expired Token	Response when a token was already redeemed.
609	Token Generation Error	Response when the voucher feature is disabled
901	Internal error	Please retry

13. Appendix B: Message statuses

These are message statuses that are generated after the Clickatell gateway has accepted the message for delivery. Data regarding messages passing initial validation and accepted for delivery will be included in your Clickatell Central reports.

Number	Hex	Description	Detail
001	0x001	Message unknown	The message ID is incorrect, or reporting is delayed.
002	0x002	Message queued	The message could not be delivered and has been queued for attempted redelivery.
003	0x003	Delivered to gateway	Delivered to the upstream gateway or network (delivered to the recipient).
004	0x004	Received by recipient	Confirmation of receipt on the handset of the recipient.
005	0x005	Error with message	There was an error with the message, probably caused by the content of the message itself.
006	0x006	User cancelled message delivery	The message was terminated by a user (stop message command) or by our staff.
007	0x007	Error delivering message	An error occurred delivering the message to the handset
008	0x008	OK	Message received by gateway.
009	0x009	Routing error	The routing gateway or network has had an error routing the message.

010	0x00A	Message expired	Message has expired before we were able to deliver it to the upstream gateway. No charge applies.
011	0x00B	Message queue for later delivery	Message has been queued at the gateway for delivery at a later time (delayed delivery).
012	0x00C	Out of credit	The message cannot be delivered due to a lack of funds in your account. Please re-purchase credits.
014	0x00E	Maximum MT limit exceeded	The allowable amount for MT messaging has been exceeded.

14. Appendix C: XML DTD

Below is the DTD for the XML API. However, it is recommended that you download the latest DTD from http://www.clickatell.com/downloads/xml_dtd.txt

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT clickAPI (auth?, sendMsg?, queryMsg?, delMsg?, ping?, getBalance?, startBatch?, sendItem?,
quickSend?, endBatch?, sendOTA?, authResp?, sendMsgResp?, queryMsgResp?, delMsgResp?, pingResp?,
getBalanceResp?, startBatchResp?, sendItemResp?, quickSendResp?, endBatchResp?, sendOTAResp?)>
<!ELEMENT auth ((api_id, user, password), sequence_no?)>
<!ELEMENT authResp ((session_id | fault), sequence_no?)>
<!ELEMENT sendMsg (((api_id, user, password) | session_id), concat?, callback?, cliMsgId?,
deliv_ack?, deliv_time?, escalate?, from?, max_credits?, msg_type?, queue?, req_feat?, text, to,
udh?, unicode?, validity?, sequence_no?)>
<!ELEMENT sendMsgResp ((apiMsgId | fault), sequence_no?)>
<!ELEMENT delMsg (((api_id, user, password) | session_id), (cliMsgId | apiMsgId), sequence_no?)>
<!ELEMENT delMsgResp (((apiMsgId, status) | fault), sequence_no?)>
<!ELEMENT getBalance (((api_id, user, password) | session_id), sequence_no?)>
<!ELEMENT getBalanceResp ((ok | fault), sequence_no?)>

<!ELEMENT ping (session_id, sequence_no?)>
<!ELEMENT pingResp ((ok | fault), sequence_no?)>
<!ELEMENT tokenPay (session_id, token, sequence_no?)>
<!ELEMENT tokenPayResp ((ok | fault), sequence_no?)>
<!ELEMENT routeCoverage (msisdn, sequence_no?)>
<!ELEMENT routeCoverageResp ((ok | fault), sequence_no?)>
<!ELEMENT queryMsg (((api_id, user, password) | session_id), (cliMsgId | apiMsgId), sequence_no?)>
<!ELEMENT queryMsgResp (((apiMsgId, status) | fault), sequence_no?)>
<!ELEMENT startBatch (((api_id, user, password) | session_id), concat?, callback?, cliMsgId?,
deliv_ack?, deliv_time?, escalate?, from?, max_credits?, msg_type?, queue?, req_feat?, template,
udh?, unicode?, validity?, sequence_no?)>
<!ELEMENT startBatchResp ((batch_id | fault), sequence_no?)>
<!ELEMENT quickSend (((api_id, user, password) | session_id), batch_id, to, sequence_no?)>
<!ELEMENT quickSendResp ((apiMsgId | fault), sequence_no?)>
<!ELEMENT sendItem (((api_id, user, password) | session_id), batch_id, to, fields?, sequence_no?)>
<!ELEMENT sendItemResp ((apiMsgId | fault), sequence_no?)>
<!ELEMENT endBatch (((api_id, user, password) | session_id), batch_id, sequence_no?)>
<!ELEMENT endBatchResp ((ok | fault), sequence_no?)>
<!ELEMENT sendOTA (((api_id, user, password) | session_id), concat?, callback?, cliMsgId?,
deliv_ack?, deliv_time?, validity?, from?, to, ota_type, name?, url?, bearer?, ppp_logintype?,
ppp_authtype?, ppp_authname?, ppp_authsecret?, proxy?, csd_dialstring?, csd_callspeed?,
csd_calltype?, proxy_type?, proxy_logintype?, proxy_authname?, proxy_authsecret?, port?, isp_name?,
sms_smsc?, gprs_access?, xml?, sequence_no?)>
<!ELEMENT sendOTAResp ((apiMsgId | fault), sequence_no?)>
<!ELEMENT fields (#PCDATA | field1 | field2 | field3 | field4 | field5 | field6 | field7 | field8 |
field9 | field10)*>
<!ELEMENT api_id (#PCDATA)>
```

```
<!ELEMENT apiMsgId (#PCDATA)>
<!ELEMENT batch_id (#PCDATA)>
<!ELEMENT bearer (#PCDATA)>
<!ELEMENT callback (#PCDATA)>
<!ELEMENT cliMsgId (#PCDATA)>
<!ELEMENT concat (#PCDATA)>
<!ELEMENT csd callspeed (#PCDATA)>
<!ELEMENT csd calltype  (#PCDATA)>
<!ELEMENT csd dialstring  (#PCDATA)>
<!ELEMENT deliv_ack (#PCDATA)>
<!ELEMENT deliv time (#PCDATA)>
<!ELEMENT escalate (#PCDATA)>
<!ELEMENT fault (#PCDATA)>
<!ELEMENT field1 (#PCDATA)>
<!ELEMENT field2 (#PCDATA)>
<!ELEMENT field3 (#PCDATA)>
<!ELEMENT field4 (#PCDATA)>
<!ELEMENT field5 (#PCDATA)>
<!ELEMENT field6 (#PCDATA)>
<!ELEMENT field7 (#PCDATA)>
<!ELEMENT field8 (#PCDATA)>
<!ELEMENT field9 (#PCDATA)>
<!ELEMENT field10 (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT gprs_access (#PCDATA)>
<!ELEMENT isp name (#PCDATA)>
<!ELEMENT max credits (#PCDATA)>
<!ELEMENT msg type (#PCDATA)>
<!ELEMENT name  (#PCDATA)>
<!ELEMENT ok (#PCDATA)>
<!ELEMENT ota_type (#PCDATA)>
<!ELEMENT password (#PCDATA)>
<!ELEMENT port (#PCDATA)>
<!ELEMENT ppp_authname (#PCDATA)>
<!ELEMENT ppp_authsecret (#PCDATA)>
<!ELEMENT ppp authtype (#PCDATA)>
<!ELEMENT ppp logintype  (#PCDATA)>
<!ELEMENT proxy (#PCDATA)>
<!ELEMENT proxy_authname (#PCDATA)>
<!ELEMENT proxy authsecret (#PCDATA)>
<!ELEMENT proxy_logintype (#PCDATA)>
<!ELEMENT proxy type (#PCDATA)>
<!ELEMENT queue (#PCDATA)>
<!ELEMENT req_feat (#PCDATA)>
<!ELEMENT sequence_no (#PCDATA)>
<!ELEMENT session id (#PCDATA)>
<!ELEMENT sms smsc (#PCDATA)>
<!ELEMENT status (#PCDATA)>
<!ELEMENT text (#PCDATA)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT udh (#PCDATA)>
<!ELEMENT unicode (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT user (#PCDATA)>
<!ELEMENT validity (#PCDATA)>
<!ELEMENT xml (#PCDATA)>
```

15. Appendix D: XML references

A reference allows you to include additional text or markup in an XML document. References always begin with the character “&” (which is specially reserved) and end with the character “;”.

XML has two kinds of references:

15.1 Entity references

An entity reference, like “&”, contains a name (in this case, “amp”) between the start and end delimiters. The name refers to a predefined string of text and/or markup, like a macro in the C or C++ programming languages.

15.2 Character references

A character reference, like "&", contains a hash mark ("#"), followed by a number. The number always refers to the Unicode code for a single character, such as 65 for the letter "A" or 233 for the letter "é", or 8211 for an en-dash.

For advanced uses, XML provides a mechanism for declaring your own entities, but that is outside the scope of this document. XML also provides five pre-declared entities that you can use to escape (from using?) special characters in an XML document:

Character	Predeclared Entity
&	&amp;
<	&lt;
>	&gt;
"	&quot;
'	&apos;

For example, the corporate name "AT&T" should appear in the XML markup as "AT&T": the XML parser will take care of changing "&" back to "&" automatically when the document is processed.

You must use character references for Greek and other extended characters.

Δ ΦΓ Λ ΩΠ ΨΣ ΘΞ

Becomes
&#x81;&#x82;&#x83;&#x84;&#x85;&#x86;&#x87;&#x88;&#x89;

For correct delivery of Greek uppercase characters, please contact our support team for more information on using a Greek routing profile.

16. Terminology

- **Receiving Messages:** A message sent (originating) from a mobile handset to an application via Clickatell.
- **Sending Messages:** A message sent from an application to (terminating on) a mobile handset via Clickatell.
- **Premium rated message:** A mobile user is charged a premium for the message that they send to a particular short or long code. This service is not available in all regions; please contact an Account Manager for more information.

- **Revenue share:** This refers to the portion of the premium charge associated with a premium rated message, which is passed on to the content provider.
- **Content provider:** This is the Clickatell customer who is offering one or more services that are usually premium rated SMS system.
- **Customer:** A registered Clickatell customer utilizing the Clickatell API for message delivery and receipt.
- **Sender ID:** The "from" address that appears on the user's handset. This is also known as the message originator or source address. A Sender ID must be registered within your account and approved by us before it may be used.
- **Destination address:** The mobile number/MSISDN of the handset to which the message must be delivered. The number should be in international number format, e.g., country code + local mobile number, excluding the leading zero (0).
- **Source address:** See 'Sender ID' above.
- **Short code:** A short number which is common across all the operators for a specific region.
- **Subscriber:** The mobile network subscriber who owns the mobile number (MSISDN) which will send or receive SMSs or be billed for premium rated services.
- **Upstream gateway:** A network operator, third party or our own short message service center (SMSC).

17. Contact details

Phone: +27 21 910 7700
Fax: +27 21 910 7701
Website: www.clickatell.com
Help URL: https://www.clickatell.com/about-us/contact-us/contact-support/
Support: support@clickatell.com
Info: info@clickatell.com
Sales: sales@clickatell.com