



Dialogue RESTful SMS API

API Developer Guide

Date: March 2013

Version: 1.6.2

CONTENT

1	Introduction.....	3
1.1	Authentication	3
1.2	Testing	4
1.2.1	Curl.....	4
1.2.2	Browser Add-on	5
2	Message Submission.....	6
2.1	XML Format	7
2.2	JSON Format	8
2.3	Form Data Format.....	9
2.4	Recipient Formatting	9
2.5	Character Support.....	9
2.6	Long Message Concatenation.....	11
2.7	Scheduling	13
2.8	Delivery Reports.....	14
2.9	Two-way Messaging	18
2.10	Message Tagging	21
2.11	Message Validity.....	22
2.12	Status Codes	23
2.12.1	Transaction Completed (00 to 1F).....	23
2.12.2	Permanent Errors (40 to 5F)	23
2.12.3	Retry Errors (60 to 7F)	24

1 Introduction

The Dialogue RESTful SMS API allows you to make use of Dialogue's SMS services and to connect these to your own or third-party applications.

The API is implemented as a RESTful web service, which is accessed over HTTP using XML or JSON. The API follows the REST principles and is easy to use: you perform a GET request to read data, a PUT request to create or update data or a DELETE request to delete data. In RESTful services a POST request is typically used to create a new unique entry (such as a blog entry). The API uses POST requests to submit SMS messages. For more information on REST see <http://en.wikipedia.org/wiki/REST>.

The REST API is language-neutral; it can be accessed from any language or platform that supports HTTP 1.1 requests. However, if you are a C#, Java or PHP developer we suggest you look at the Client Libraries and related Quick Start Guides. The client library creates a wrapper around the REST protocol allowing you to send a message in just a few lines of code. The client library also provides utility classes that help you to receive and process delivery reports and reply messages.

Language/Platform	Quick Start Guide
.NET (*)	Dialogue SMS REST API – C# Quick Start Guide
Java	Dialogue SMS REST API – Java Quick Start Guide
PHP	Dialogue SMS REST API – PHP Quick Start Guide

(*) The code samples in the .NET Quick Start Guide target C# developers; however, the client library can be used with any .NET compatible language such as Visual Basic or C++.

You will need to reference the relevant Configuration Document to learn how to get your credentials and to configure the calls in respect to your account to access the Dialogue API and to send messages.

1.1 Authentication

As part of the sign-up process you will receive your account credentials consisting of a user name and a password (see the relevant configuration document for more details). These credentials are used to access the REST API. Every API call issued requires the user name and password to be sent via HTTP Basic Access authentication (see http://en.wikipedia.org/wiki/Basic_access_authentication). Some platforms such as .NET allow setting a Credentials object on the HTTP request; others require you to manually add the required HTTP header.

The HTTP header to be set is:

Authorization: Basic <authInfo>

where `<authInfo>` is `<userName>:<password>` (user name and password separated by a colon) encoded using the Base64 algorithm. You don't need to worry about the details of the encoding algorithm as most platforms will provide you with some means of performing this encoding automatically. If you want to know more about Base64 please see <http://en.wikipedia.org/wiki/Base64>.

For example, if the user name were "myuser" and the password "mypass" you would first concatenate these two separated by a colon, e.g. "myuser:mypass". This string is then encoded using Base64 and becomes "bXl1c2VyOm15cGFzcw==". The HTTP header is therefore set to:

Authorization: Basic bXl1c2VyOm15cGFzcw==

The code samples illustrate how to perform HTTP requests with authentication.



Basic Access authentication on its own is not secure. Therefore we strongly recommend that you access the API via SSL, i.e. using `https://...` instead of `http://...`

1.2 Testing

To perform simple language-independent tests we suggest two tools that can be used to access the API:

1.2.1 Curl

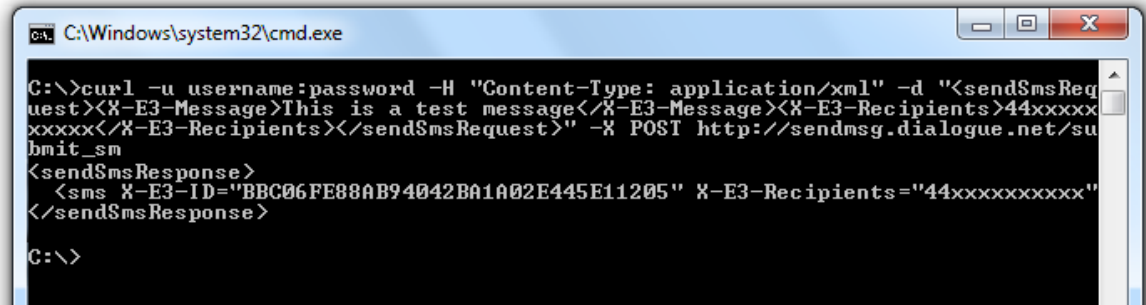
Curl (see <http://curl.haxx.se/>) is a command line tool available on many systems for performing HTTP requests. Using the `-u` option you can specify user name and password (Curl automatically adds the required Base64-encoded header), the `-H` option specifies the content type, e.g. `application/xml` or `application/json`, using the `-d` option you can pass the content itself (the XML or JSON payload) and using the `-X` option you can tell it what method to use (GET, POST, PUT, etc.).

For example, the command below can be used to perform a message submission as XML (substitute the values marked yellow with your own credentials and recipient number):

```
curl -u username:password -H "Content-Type: application/xml" -d
"<sendSmsRequest><X-E3-Message>This is a test message</X-E3-Message><X-E3-
Recipients>44xxxxxxxxxx</X-E3-Recipients></sendSmsRequest>" -X POST
http://sms.dialogue.net/submit_sm
```

If the submission was successful the response curl prints should look like this:

```
<sendSmsResponse>
  <sms X-E3-ID="01F4AC40DAA84143A3D6BAE3D139BB51" X-E3-
Recipients="44xxxxxxxxxx" X-E3-Submission-Report="00" />
</sendSmsResponse>
```



```
C:\Windows\system32\cmd.exe
C:\>curl -u username:password -H "Content-Type: application/xml" -d "<sendSmsRequest><X-E3-Message>This is a test message</X-E3-Message><X-E3-Recipients>44xxxxxxx</X-E3-Recipients></sendSmsRequest>" -X POST http://sendmsg.dialogue.net/submit_sms
<sendSmsResponse>
  <sms X-E3-ID="BBC06FE88AB94042BA1A02E445E11205" X-E3-Recipients="44xxxxxxxxxx">
</sendSmsResponse>
C:\>
```

1.2.2 Browser Add-on

Browser-based REST client add-ons exist for Firefox and Chrome. We'll focus on the Firefox add-on here, which can be installed from <http://addons.mozilla.org/en-US/firefox/addon/restclient/>.

When installed select **REST Client** from the browser's Tools menu; this opens the client in a new tab. To perform a message submission as XML, follow these steps:

1. Select POST from the **Method** drop down
2. Set the URL (next to **Method** drop down) to the relevant end point.
3. Click the **Login** button, leave the **Basic** option selected and enter your credentials (user name and password); this adds the necessary encoded Authorization header
4. Click **Add Request Header** add enter Content-Type under **Name** and application/xml under **Value** (be careful not to include any trailing spaces when copying and pasting the header's name or value)
5. Provide the following XML as **Request Body** (substitute 44xxxxxxxxxx with your own recipient number):

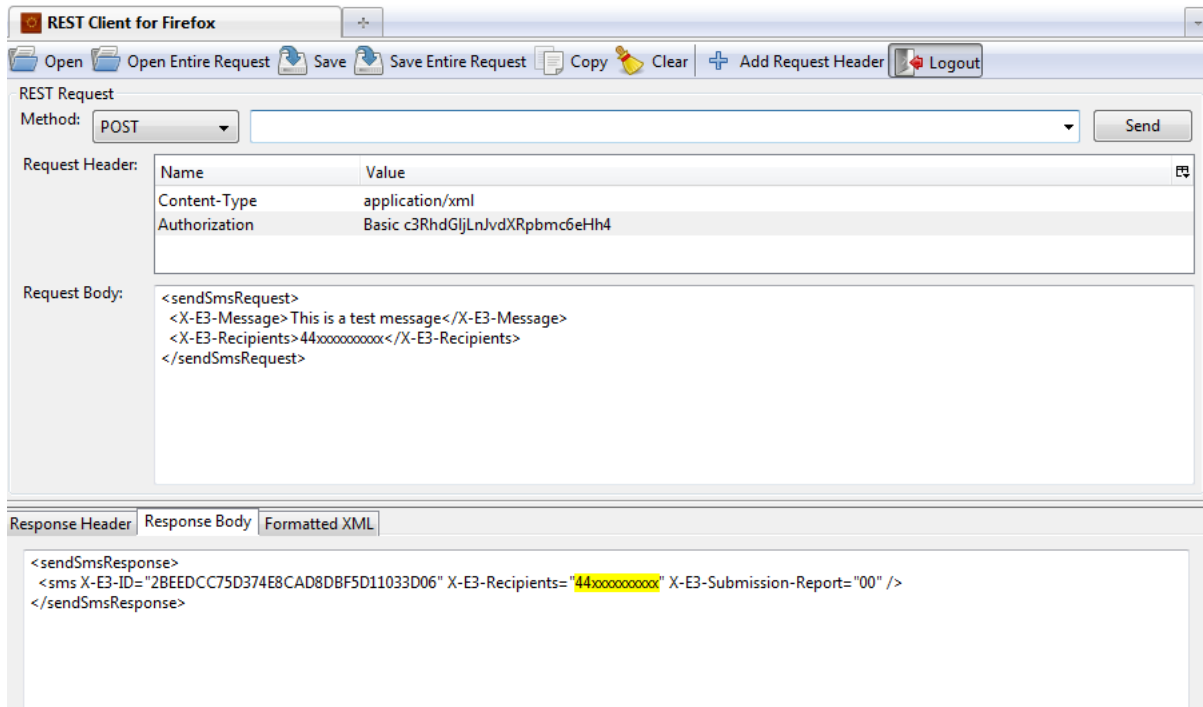
```
<sendSmsRequest>
  <X-E3-Message>This is a test message</X-E3-Message>
  <X-E3-Recipients>44xxxxxxxxxx</X-E3-Recipients>
</sendSmsRequest>
```

6. Click the **Send** button
7. Select the **Response Body** tab

If the submission was successful the response should look like this:

```
<sendSmsResponse>
  <sms X-E3-ID="2BEEDCC75D374E8CAD8DBF5D11033D06" X-E3-Recipients="44xxxxxxxxxx" X-E3-Submission-Report="00" />
</sendSmsResponse>
```

Note: The end point below has been removed you will need to refer to the configuration guide to get this.



The screenshot shows the 'REST Client for Firefox' window. The 'Method' is set to 'POST'. The 'Request Header' section contains a table with the following data:

Name	Value
Content-Type	application/xml
Authorization	Basic c3RhZGJjLnVvdXRpbmc6eHh4

The 'Request Body' contains the following XML:

```
<sendSmsRequest>
  <X-E3-Message>This is a test message</X-E3-Message>
  <X-E3-Recipients>44xxxxxxxx</X-E3-Recipients>
</sendSmsRequest>
```

The 'Response Body' tab is selected, showing the following XML response:

```
<sendSmsResponse>
  <sms X-E3-ID="2BEEDCC75D374E8CAD8DBF5D11033D06" X-E3-Recipients="44xxxxxxxx" X-E3-Submission-Report="00" />
</sendSmsResponse>
```

2 Message Submission

We recommend that you use the encrypted, secure URL of https rather than unencrypted http.

The required request method is POST, which includes a number of name/value parameters. Mandatory parameters are X-E3-Message (defines the message text) and X-E3-Recipients (defines the recipient).

Here an overview of the basic parameters used to send a message:

Parameter name	Type	Required	Maximum length	Description
X-E3-Message	Multiple	Yes	160 †	Message text(s)
X-E3-Recipients	Multiple	Yes	Depends on numbering scheme	Recipient number(s)

† Maximum length depends on the characters used inside the message text and whether concatenation is enabled or not.

The API allows the submission of multiple messages to multiple recipients; X-E3-Message and X-E3-Recipients parameters may occur several times when sending multiple messages and/or to target multiple recipients, respectively. In XML format these parameters are included as multiple XML elements (see section 0); in JSON format these are provided as arrays (see section 2.2).

The response includes a collection of submission results, one for each message whether it was sent successfully or not. Each of these results includes the fields X-E3-Recipients and X-E3-Submission-Report. X-E3-Recipients identifies the recipient (passed in the request). X-E3-Submission-Report indicates whether the message was sent successfully. If the submission was successful the value will be 00, otherwise it indicates the error status code (status codes and their meaning are listed in section 2.12).

If the submission succeeds, the result will include the X-E3-ID field. This contains a unique submission identifier and can be used to track message delivery. If the submission fails, the result will include X-E3-Error-Description containing an error description. This corresponds to the status code in X-E3-Submission-Report.

The order of parameters and fields in both requests and responses is not important and no assumptions should be made regarding the order when parsing the response. This applies to both XML and JSON formats. Similarly, any unknown fields in the response should be ignored since additional fields may be added in the future.

2.1 XML Format

Provide an HTTP header Content-Type: application/xml or Content-Type: application/xml; charset=UTF-8 (recommended).

Request format:

```
<sendSmsRequest>
  <X-E3-Message>This is a test message</X-E3-Message>
  <X-E3-Message>This is another message</X-E3-Message>
  <X-E3-Recipients>44xxxxxxxxxx</X-E3-Recipients>
  <X-E3-Recipients>44xxxxxxxxxx</X-E3-Recipients>
```

```
</sendSmsRequest>
```

Response format:

```
<sendSmsResponse>
  <sms X-E3-ID="90A9893BC2B645918034F4C358A062CE" X-E3-
Recipients="44xxxxxxxxxx" X-E3-Submission-Report="00" />
  <sms X-E3-ID="2BEEDCC75D374E8CAD8DBF5D11033D06" X-E3-
Recipients="44xxxxxxxxxx" X-E3-Submission-Report="00" />
  <sms X-E3-Error-Description="Something went wrong" X-E3-
Recipients="44xxxxxxxxxx" X-E3-Submission-Report="45" />
  <sms X-E3-Error-Description="Something went wrong" X-E3-
Recipients="44xxxxxxxxxx" X-E3-Submission-Report="45" />
</sendSmsResponse>
```

2.2 JSON Format

Provide an HTTP header Content-Type: application/json or Content-Type: application/json; charset=UTF-8 (recommended).

Request format:

```
{
  "X-E3-Message": ["This is a test message", "This is another message"],
  "X-E3-Recipients": ["44xxxxxxxxxx", "44xxxxxxxxxx"]
}
```

Response format:

```
{
  "sms": [
    {
      "X-E3-Recipients": "44xxxxxxxxxx",
      "X-E3-Submission-Report": "00",
      "X-E3-ID": "90A9893BC2B645918034F4C358A062CE"
    },
    {
      "X-E3-Recipients": "44xxxxxxxxxx",
      "X-E3-Submission-Report": "00",
      "X-E3-ID": "2BEEDCC75D374E8CAD8DBF5D11033D06"
    },
    {
      "X-E3-Error-Description": "Something went wrong",
      "X-E3-Recipients": "44xxxxxxxxxx",
      "X-E3-Submission-Report": "45"
    },
    {
      "X-E3-Error-Description": "Something went wrong",
```



```

        "X-E3-Recipients": "44xxxxxxxxxx",
        "X-E3-Submission-Report": "45"
    }
]
}

```

2.3 Form Data Format

Provide an HTTP header Content-Type: application/x-www-form-urlencoded or Content-Type: application/x-www-form-urlencoded; charset=UTF-8 (recommended). Each of the parameters is specified as a HTTP FORM parameter, either as a HTTP POST request (recommended) or HTTP GET.

Note if you are manually creating the request body you must correctly URL-encode each parameter, e.g. spaces become %20, etc. Most languages and platforms provide the means to perform URL-encoding.

Request format:

```

X-E3-Message=This%20is%20a%20test%20message&X-E3-
Message=This%20is%20another%20message&X-E3-
Recipients=44xxxxxxxxxx&X-E3-Recipients=44xxxxxxxxxx

```

Response format (legacy, deprecated formatting in grey):

```

<!--
E3_SUBMIT_RESULT=44xxxxxxxxxx:success:5C3F3CD88D504801986C8C45C38B30
40 -->
<!-- X-E3-Recipients: "44xxxxxxxxxx" X-E3-Submission-Report: "00" X-
E3-ID: "5C3F3CD88D504801986C8C45C38B3040" -->

```

2.4 Recipient Formatting

Recipient numbers must be written in normalized international form. Numbers must start with the country's international dialling prefix, such as 1 for the United States or 44 for United Kingdom, followed by the recipient's number. Do not prefix the number with + or 00. Also, do not include trunk prefixes used in some countries, e.g. the UK mobile numbering starts with 07... but in international form this becomes 447...

Recipient numbers must not include any characters except digits, i.e. do not include spaces, brackets, letters or symbols.

2.5 Character Support

Generally we recommend you submit all messages using the universal UTF-8 character set.

When using UTF-8 you must tell the API you're doing so by appending ; charset=UTF-8 to the content type header, e.g. application/xml; charset=UTF-8. When submitting in XML format you must also indicate UTF-8 encoding in the XML header: <?xml version="1.0" encoding="UTF-8" ?>

Here is an example:

```
POST /submit_sm HTTP/1.1
Content-Type: application/xml; charset=UTF-8
Authorization: Basic azhqewtjMjo0Zjl2aXFkaA==
Host: sms.dialogue.net
Content-Length: 264

<?xml version="1.0" encoding="UTF-8" ?>
<sendSmsRequest>
  <X-E3-Message>This is a test message</X-E3-Message>
  <X-E3-Recipients>44xxxxxxxxxx</X-E3-Recipients>
</sendSmsRequest>
```

SMS messages are sent using the GSM 03.38 alphabet, which contains these characters:

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

There are a few special GSM characters that occupy two character places; using any of these will reduce your 160 character limit:

^	{	}	\	[~]		€
---	---	---	---	---	---	---	--	---

For example, if your message starts with a Euro sign (€) you only have 158 characters left after the Euro sign, not 159 as one might expect.

Characters not listed above, such as certain accented characters (one example would be the Spanish ó) or characters from other languages such as Russian (Cyrillic) will not display correctly on the phone.

There are some submission routes that use the ISO-8859-15 character set (this is also known as Latin9). Characters marked red in the table above, specifically all the Greek characters and the inverted exclamation mark, are not represented in ISO-8859-15/Latin9 and you will not be able to send these characters. If the submission route supports the full GSM character set largely depends on the recipient's operator. You are welcome to contact your account manager to enquire about which routes support what characters. It may also be possible to configure your account to submit messages via a different route.

If you are sending static content check that your message text does not contain unsupported characters. A typical example of this is when pasting text from Microsoft Word. Word can change straight quotes into curly quotes, e.g. <"> becomes <"> or <">, which are not present in the GSM character set.

In case you are using the C/c-cedilla character (Ç or ç) you should make sure you use the lower-case version, which is part of the default GSM character set. You can also map the uppercase version to lowercase. The original GSM specification listed the letter as uppercase but the later ETSI standardisation added a correction, which changed the letter from capital C-cedilla (Ç) to small c-cedilla (ç). Therefore submitting the uppercase version will not deliver this character correctly.

2.6 Long Message Concatenation

By default any message longer than 160 characters is truncated after 160 characters.

To be able to submit a single message longer than 160 characters you can enable long message concatenation. This submits multiple message segments as parts of a "package". When all segments are received by the phone it "concatenates", i.e. puts together, all the parts into a single message that "appears" to be longer than usual.

Submitting such a long message with concatenation enabled can therefore trigger a number of individual SMS submissions, all of which will be billed to your account or use up prepay credit. Therefore, when enabling long message concatenation you are asked to provide the maximum number of allowed segments; should the long message not fit into the specified number of segments the message is truncated after the last segment.

You can enable concatenation by supplying parameter X-E3-Concatenation-Limit=**N**, where **N** is the maximum number of message segments allowed for concatenation (the

theoretical maximum being 255). For instance, to submit a concatenated message of up to 10 segments you would submit:

XML format:

```
<sendSmsRequest>
  <X-E3-Message>This is a long, long, long, long, long, long, long,
long, long, long, long, long, long, long, long, long, long, long, long,
long, long, long, long, long, long, long, long, long, long, long, long,
long test message</X-E3-Message>
  <X-E3-Recipients>44xxxxxxxxxx</X-E3-Recipients>
  <X-E3-Concatenation-Limit>10</X-E3-Concatenation-Limit>
</sendSmsRequest>
```

JSON format:

```
{
  "X-E3-Message": ["This is a long, long, long, long, long, long,
long, long, long, long, long, long, long, long, long, long,
long, long, long, long, long, long, long, long, long, long, long,
long, long, long, long, long, long, long, long, long, long, long,
long, long test message"],
  "X-E3-Recipients": ["44xxxxxxxxxx"],
  "X-E3-Concatenation-Limit": 10
}
```

Form Data format:

```
X-E3-
Message=This%20is%20a%20long%2C%20long%2C%20long%2C%20long%2C%20long
%2C%20long%2C%20long%2C%20long%2C%20long%2C%20long%2C%20long%2C%20lo
ng%2C%20long%2C%20long%2C%20long%2C%20long%2C%20long%2C%20long%2C%20
long%2C%20long%2C%20long%2C%20long%2C%20long%2C%20long%2C%20long%2C%
20long%2C%20long%2C%20long%2C%20long%2C%20long%2C%20long%2C%20long%2
C%20long%2C%20long%2C%20long%2C%20long%2C%20long%2C%20long%2C%20long
%2C%20long%2C%20long%20test%20message&X-E3-
Recipients=44xxxxxxxxxxx&X-E3-Concatenation-Limit=10
```

Even if the long message has only 1 recipient the responses may contain several entries, for example:

XML response:

```
<sendSmsResponse>  
  <sms X-E3-ID="473B827B78774DB18D90D4C6D9FE17A8" X-E3-  
Recipients="44xxxxxxxxxxx" X-E3-Submission-Report="00" />  
  <sms X-E3-ID="4CE0184C3C354ACF8F5A70F457242C6F" X-E3-  
Recipients="44xxxxxxxxxxx" X-E3-Submission-Report="00" />
```

```
</sendSmsResponse>
```

JSON response:

```
{
  "sms" : [
    {
      "X-E3-Recipients" : "44xxxxxxxxxx",
      "X-E3-Submission-Report" : "00",
      "X-E3-ID" : "89E1EE59824A4EB5B4B7CA693B6C69D2"
    },
    {
      "X-E3-Recipients" : "44xxxxxxxxxx",
      "X-E3-Submission-Report" : "00",
      "X-E3-ID" : "F08BBEF1CA284E5180F61D3786D44395"
    }
  ]
}
```

Form Data response (legacy, deprecated formatting in grey):

```
<!--
E3_SUBMIT_RESULT=44xxxxxxxxxx:success:59944627B250458CB20AAF324148F4
75 -->
<!-- X-E3-Recipients: "44xxxxxxxxxx" X-E3-Submission-Report: "00" X-
E3-ID: "59944627B250458CB20AAF324148F475" -->
<!--
E3_SUBMIT_RESULT=44xxxxxxxxxx:success:C1795D2427AC4E128AD422BE5E791F
56 -->
<!-- X-E3-Recipients: "44xxxxxxxxxx" X-E3-Submission-Report: "00" X-
E3-ID: "C1795D2427AC4E128AD422BE5E791F56" -->
```

If used, concatenation brings with it an extra fragmentation overhead in each of the message segments. This leaves less characters in the segments available for the message. So sending a text message of 2 x 160, i.e. 320 characters with concatenation will need 3 segments rather than 2 segments.

Be aware that concatenation is not supported in all territories or by all network operators. If concatenation is enabled but not supported, the messaging gateway will send the message text over separate, individual messages and each message but the last one terminated by ellipsis (...).

2.7 Scheduling

You can schedule message submissions for later delivery by using the parameter X-E3-Schedule-For. The expected format is YYYYMMDDHHMMSS where:

- YYYY is the year, e.g. 2011

- *MM* is the month (00 – 12)
- *DD* is the day of the month (00 – 31)
- *HH* is the hour of the day (00 – 23)
- *MM* is the minute of the hour (00 – 59)
- *SS* is the seconds (00 – 59)

For instance, to schedule a message for submission on 1st November 2011 at 13:30:00 you would submit:

XML format:

```
<sendSmsRequest>
  <X-E3-Message>This is a test message</X-E3-Message>
  <X-E3-Recipients>44xxxxxxxxxx</X-E3-Recipients>
  <X-E3-Schedule-For>20111101133000</X-E3-Schedule-For>
</sendSmsRequest>
```

JSON format:

```
{
  "X-E3-Message": ["This is a test message"],
  "X-E3-Recipients": ["44xxxxxxxxxx"],
  "X-E3-Schedule-For": "20111101133000"
}
```

Form Data format:

```
X-E3-Message=This%20is%20a%20test%20message&X-E3-
Recipients=44xxxxxxxxxx&X-E3-Schedule-For=20111101133000
```

Note that scheduling message submissions in the past will send the message(s) immediately.

The default time zone of X-E3-Schedule-For is Europe/London. Additionally NN+ or NN- can be appended to X-E3-Schedule-For, where NN is the time difference in 15 minute intervals, i.e. 04 for 1 hour, 08 for 2 hours, etc., to offset from the default time zone.



Note that the NN+/NN- offset is relative to the Europe/London time zone and not UTC, meaning you need to take into account DST (Daylight Saving Time) changes (GMT vs. BST) when scheduling message submissions.

2.8 Delivery Reports

You can enable delivery reports by using the parameter X-E3-Confirm-Delivery=on, and specifying a reply path using X-E3-Reply-Path to point to a call-back handler that you need to implement. This handler is invoked via HTTP or HTTPS when a message is confirmed as delivered. It may also be invoked for temporary delivery failures and will be invoked for

permanent delivery failures, e.g. after all retry attempts have been exhausted. The reply path must be formatted as:

```
http://www.mysite.com/path
```

or

```
https://www.mysite.com/path
```

You can optionally provide a custom or third-party identifier using the parameter X-E3-User-Key. The user key is included in the delivery report allowing you to uniquely identify a report and submission. For instance the following example shows how to use delivery confirmations:

XML format:

```
<sendSmsRequest>
  <X-E3-Message>This is a test message</X-E3-Message>
  <X-E3-Recipients>44xxxxxxxxxx</X-E3-Recipients>
  <X-E3-Confirm-Delivery>on</X-E3-Confirm-Delivery>
  <X-E3-Reply-Path>http://www.mysite.com/path</X-E3-Reply-Path>
  <X-E3-User-Key>1234567890</X-E3-User-Key>
</sendSmsRequest>
```

JSON format:

```
{
  "X-E3-Message": ["This is a test message"],
  "X-E3-Recipients": ["44xxxxxxxxxx"],
  "X-E3-Confirm-Delivery": "on",
  "X-E3-Reply-Path": "http://www.mysite.com/path",
  "X-E3-User-Key": "1234567890"
}
```

Form Data format:

```
X-E3-Message=This%20is%20a%20test%20message&X-E3-
Recipients=44xxxxxxxxxx&X-E3-Confirm-Delivery=on&X-E3-Reply-
Path=http%3A%2F%2Fwww.mysite.com%2Fpath&X-E3-User-Key=1234567890
```

The call-back handler specified by X-E3-Reply-Path must support HTTP POST requests in application/xml, application/json or application/x-www-form-urlencoded formats, depending on which format was used during submission.

The application/x-www-form-urlencoded format is supported by nearly all web servers. You can get named parameters from the request object, for example, request.getParameter("X-E3-...") in Java/JSP or Request["X-E3-..."] in ASP.NET.

The following parameters are available:

- **X-E3-Recipients**

The mobile number to which the message was sent (X-E3-Recipients in submission request).

- **X-E3-ID**

The unique submission identifier that relates the delivery report to the submission (X-E3-ID in submission response).

X-E3-User-Key

Only if applicable. The user key used during the submission (X-E3-User-Key in submission request). This can be used to uniquely associate submissions with reports.

- **X-E3-Delivery-Report**

Status code; “00” indicates the message was successfully delivered (for a full list of possible codes see section 2.12.). You must distinguish between temporary failure codes, which can still be retried, and permanent failure codes, which will not be retried. Do not make the assumption that anything other than “00” is automatically a failed delivery.

- **X-E3-Timestamp**

The time the report entered Dialogue’s system formatted as YYYY-MM-DD HH:MM:SS, where:

- YYYY is the year, e.g. 2011
- MM is the month (00 – 12)
- DD is the day of the month (00 – 31)
- HH is the hour of the day (00 – 23)
- MM is the minute of the hour (00 – 59)
- SS is the seconds (00 – 59)

The time zone X-E3-Timestamp uses is Europe/London.

- **X-E3-Network**

The recipient’s network operator name.

Here are some example posts of what you might expect:

XML format:

```
POST /path HTTP/1.1
From: sms-master@uk.dialogue.net
Host: www.mysite.com
Content-Type: application/xml
```


Content-Length: 287
User-Agent: E3 sms2http

```
<callback X-E3-Delivery-Report="00" X-E3-
ID="3783B261BCEC4808B6DFC5E116849DEC" X-E3-Loop="1321535755.2806"
X-E3-Network="Orange" X-E3-Recipients="44xxxxxxxxxx" X-E3-
Timestamp="2011-11-17 13:15:54" X-E3-User-Key="1234567890" />
```

JSON format:

```
POST /path HTTP/1.1
From: sms-master@uk.dialogue.net
Host: www.mysite.com
Content-Type: application/json
Content-Length: 287
User-Agent: E3 sms2http
```

```
{
  "X-E3-Recipients" : "44xxxxxxxxxx",
  "X-E3-Network" : "Orange",
  "X-E3-ID" : "3783B261BCEC4808B6DFC5E116849DEC",
  "X-E3-Loop" : 1321535755.2806,
  "X-E3-User-Key" : "1234567890",
  "X-E3-Timestamp" : "2011-11-17 13:15:54",
  "X-E3-Delivery-Report" : "00"
}
```

Form Data format:

```
POST /path HTTP/1.1
From: sms-master@uk.dialogue.net
Host: www.mysite.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 244
User-Agent: E3 sms2http
```

```
X%2dE3%2dRecipients=44xxxxxxxxxx&X%2dE3%2dNetwork=Orange&X%2dE3%2dID
=3783B261BCEC4808B6DFC5E116849DEC&X%2dE3%2dLoop=1321535755%2e2806&X%
2dE3%2dUser%2dKey=1234567890&X%2dE3%2dTimestamp=2011%2d11%2d17%2013%
3a15%3a54&X%2dE3%2dDelivery%2dReport=00
```

To acknowledge having received the notification you must respond with an HTTP status 200 (OK). The Dialogue messaging gateway will periodically retry the request if your web server is not available or an HTTP status other than 200 was returned.

2.9 Two-way Messaging

A two-way or session-tracked message allows the recipient(s) to reply. The sender is automatically set to a specific number from a pool of virtual numbers. When a recipient replies the recipient's number and virtual number together identify the original session.

To enable session tracking you must supply a reply path using the parameter X-E3-Session-Reply-Path. You must point this path to a call-back handler that you need to implement. This handler is invoked via HTTP or HTTPS when a reply is received. The reply path must be formatted as:

`http://www.mysite.com/path`

or

`https://www.mysite.com/path`

Optionally you can specify a session identifier using the parameter X-E3-Session-ID; if this is not provided the handler will be invoked without a session identifier. The session identifier can be any arbitrary string, we suggest you generate some kind of UUID (Universally Unique Identifier) and store this for later reference. The following example shows how to use two-way messaging with a sample session identifier of "cf9c9a2a99c1474783b09cdd17ad616e":

XML format:

```
<sendSmsRequest>
  <X-E3-Message>This is a test message</X-E3-Message>
  <X-E3-Recipients>44xxxxxxxxxx</X-E3-Recipients>
  <X-E3-Session-Reply-Path>http://www.mysite.com/path</X-E3-Session-Reply-Path>
  <X-E3-Session-ID>cf9c9a2a99c1474783b09cdd17ad616e</X-E3-Session-ID>
</sendSmsRequest>
```

JSON format:

```
{
  "X-E3-Message": ["This is a test message"],
  "X-E3-Recipients": ["44xxxxxxxxxx"],
  "X-E3-Session-Reply-Path": "http://www.mysite.com/path",
  "X-E3-Session-ID": "cf9c9a2a99c1474783b09cdd17ad616e"
}
```

Form Data format:

```
X-E3-Message=This%20is%20a%20test%20message&X-E3-Recipients=44xxxxxxxxxx&X-E3-Session-Reply-Path=http%3A%2F%2Fwww.mysite.com%2Fpath&X-E3-Session-ID=cf9c9a2a99c1474783b09cdd17ad616e
```

The call-back handler specified by X-E3-Reply-Path must support HTTP POST requests in application/xml, application/json or application/x-www-form-urlencoded formats, depending on which format was used in the submission.

The application/x-www-form-urlencoded format is supported by nearly all web servers, i.e. you can get named parameters from the request object (for example, request.getParameter("X-E3-...") in Java/JSP or Request["X-E3-..."] in ASP.NET).

The following parameters are available:

- **X-E3-Originating-Address**

The mobile number to which the message was sent. This ties the reply message to the original submission. This is the X-E3-Recipients parameter in submission request.

- **X-E3-Session-ID**

Applicable if a session identifier was used during the submission (X-E3-Session-ID in the submission request). This can be used to uniquely associate submissions with replies.

- **X-E3-Hex-Message**

Hex-encoded text of the reply message in ISO-8859-15. Each byte is represented by a two-digit hexadecimal number. To convert this into a usable string first decode the hex-encoded string into an array of bytes then construct a string object from these bytes using the ISO-8859-15 (also called Latin9) character set.

- **X-E3-ID**

The unique identifier of the reply message (this is different to the identifier returned by original submission).

- **X-E3-Timestamp**

The time the reply entered Dialogue's system formatted as YYYY-MM-DD HH:MM:SS.000000, where:

- YYYY is the year, e.g. 2011
- MM is the month (00 – 12)
- DD is the day of the month (00 – 31)
- HH is the hour of the day (00 – 23)
- MM is the minute of the hour (00 – 59)
- SS is the seconds (00 – 59)

The time zone X-E3-Timestamp uses is Europe/London.

- **X-E3-Network**

The recipient's network operator name.

Here some example posts of what you might expect (less frequently used parameters are shown in grey):

XML format:

```
POST /path HTTP/1.1
From: sms-master@uk.dialogue.net
Host: www.mysite.com
Content-Type: application/xml
Content-Length: 434
User-Agent: E3 sms2http

<callback X-E3-Account-Name="..." X-E3-Data-Coding-Scheme="00" X-E3-
Hex-Message="5265706C79"
X-E3-ID="94C638D49A7844D98BACAAE366C13B85" X-E3-
Loop="1321543239.27414" X-E3-MO-Campaign="" X-E3-MO-Keyword=""
X-E3-Network="Orange" X-E3-Originating-Address="44xxxxxxxxxx" X-E3-
Protocol-Identifier="00"
X-E3-Recipients="12345" X-E3-Session-ID="12345" X-E3-
Timestamp="2011-11-17 15:20:38.000000"
X-E3-User-Data-Header-Indicator="0" />
```

JSON format:

```
POST /path HTTP/1.1
From: sms-master@uk.dialogue.net
Host: www.mysite.com
Content-Type: application/json
Content-Length: 442
User-Agent: E3 sms2http

{
  "X-E3-Account-Name" : "...",
  "X-E3-User-Data-Header-Indicator" : 0,
  "X-E3-Recipients" : "12345",
  "X-E3-Hex-Message" : "5265706C79",
  "X-E3-ID" : "94C638D49A7844D98BACAAE366C13B85",
  "X-E3-MO-Keyword" : "",
  "X-E3-Data-Coding-Scheme" : "00",
  "X-E3-Network" : "Orange",
  "X-E3-MO-Campaign" : "",
  "X-E3-Loop" : 1321543239.27414,
  "X-E3-Originating-Address" : "44xxxxxxxxxx",
  "X-E3-Session-ID" : "12345",
```

```
"X-E3-Protocol-Identifier" : "00",
  "X-E3-Timestamp" : "2011-11-17 15:20:38.000000"
}
```

Form Data format:

```
POST /path HTTP/1.1
From: sms-master@uk.dialogue.net
Host: www.mysite.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 442
User-Agent: E3 sms2http

X%2dE3%2dAccount%2dName=&X%2dE3%2dUser%2dData%2dHeader%2dIndicator=0&X%2dE3%2dRecipients=12345&X%2dE3%2dHex%2dMessage=5265706C79&X%2dE3%2dID=94C638D49A7844D98BACAAE366C13B85&X%2dE3%2dData%2dCoding%2dScheme=00&X%2dE3%2dNetwork=Orange&X%2dE3%2dLoop=1321543239%2e27414&X%2dE3%2dOriginating%2dAddress=44xxxxxxxxxx&X%2dE3%2dSession%2dID=12345&X%2dE3%2dTimestamp=2011%2d11%2d17%2015%3a20%3a38%2e000000&X%2dE3%2dProtocol%2dIdentifier=00
```

To acknowledge having received the notification you must respond with an HTTP status 200 (OK). The Dialogue messaging gateway will periodically retry the request if your web server is not available or an HTTP status other than 200 was returned.

2.10 Message Tagging

Messages can be tagged by providing the X-E3-User-Tag parameter. Tagging allows certain messages to be grouped together, e.g. messages you submit on behalf of a certain customer or application. Billing records can later be exported with subtotals for each tag, (i.e. per customer or application).

The user tag cannot contain semi-colons, or be longer than 50 characters.

XML format:

```
<sendSmsRequest>
  <X-E3-Message>This is a test message</X-E3-Message>
  <X-E3-Recipients>44xxxxxxxxxx</X-E3-Recipients>
  <X-E3-User-Tag>Application_1</X-E3-User-Tag>
</sendSmsRequest>
```

JSON format:

```
{
  "X-E3-Message": ["This is a test message"],
  "X-E3-Recipients": ["44xxxxxxxxxx"],
  "X-E3-User-Tag": "Application_1"
}
```

Form Data format:

```
X-E3-Message=This%20is%20a%20test%20message&X-E3-Recipients=44xxxxxxxxxx&X-E3-User-Tag=Application_1
```

2.11 Message Validity

The parameter X-E3-Validity-Period specifies the message validity period, e.g. how long the message will be stored in the SMSC (Short Message Service Centre) awaiting delivery before it expires. The validity period is given in minutes, hours, days or weeks (note that the maximum validity period is network dependent and may vary).

The parameter is formatted as a numeric value followed by the unit “m” for minutes, “h” for hours, “d” for days or “w” for weeks. For example:

X-E3-Validity-Period	Validity Period
5m	Message expires after 5 minutes
10h	Message expires after 10 hours
3d	Message expires after 3 days

XML format:

```
<sendSmsRequest>
  <X-E3-Message>This is a test message</X-E3-Message>
  <X-E3-Recipients>44xxxxxxxxxx</X-E3-Recipients>
  <X-E3-Validity-Period>12h</X-E3-Validity-Period>
</sendSmsRequest>
```

JSON format:

```
{
  "X-E3-Message": ["This is a test message"],
  "X-E3-Recipients": ["44xxxxxxxxxx"],
  "X-E3-Validity-Period": "12h"
}
```

Form Data format:

```
X-E3-Message=This%20is%20a%20test%20message&X-E3-Recipients=44xxxxxxxxxx&X-E3-Validity-Period=12h
```

2.12 Status Codes

The following status codes are returned as X-E3-Submission-Report in message submission responses or as X-E3-Delivery-Report in message delivery confirmation notifications. The codes below are based on the ETSI 03.40 specification. Some codes are not detailed below; at the time of writing these codes are not used but reserved for future use.

It is also worth noting that some of these codes may not be applicable to your service but have been included below for completeness.

2.12.1 Transaction Completed (00 to 1F)

00	Successful
01	Sent to SME (*) but unable to confirm
02	Replaced at the SMSC

2.12.2 Permanent Errors (40 to 5F)

Errors in this class are the direct result of the networks reporting an error back to our messaging platform. No attempts are made to retry the submission. These errors indicate a permanent problem with either the recipient (for instance, the recipient may be barred) or the message, so you should not attempt to resubmit the same message to the same recipient.

40	Remote procedure error
41	Incompatible destination
42	Connection rejected by SME (*)
43	Not obtainable
44	Quality of service not available
45	No interworking available
46	Validity period expired
47	Message deleted by originating SME
48	Message deleted by SMSC admin
49	Message does not exist
4A	Unknown subscriber
4B	SMSC error (**)
50	Maximum submission attempt reached
51	Maximum Time To Live (TTL) for message reached
52	Invalid data in message
53	Non routable (operator used rejected message)
54	Authentication failure (UCP 60 mainly could be used by login page)
55	No response from SME (*)
56	SME (*) rejected message
57	Unknown error
58	Operator bar
59	Request ID not found

5A	Premium charge routing error
5B	Service ID not provisioned
5C	Mobile user disconnected and in quarantine. Mobile user's number must be removed from databases.
5D	Validity period expired with no receipt from carrier

(*) SME = Short Message Entity, an entity which may send or receive SMS.

(**) SMSC = Short Message Service Centre

2.12.3 Retry Errors (60 to 7F)

Errors in this class are the result of our messaging platform giving up trying to submit the message to the network after having unsuccessfully retried the message submission a number of times. It is a permanent error in terms of this transaction as no more attempts are made to deliver the message, however, you are allowed to retry submitting this or other future messages to the recipient since the cause of the error may be temporary.

60	Congestion
61	SME Busy
62	No response from SME (*)
63	Service rejected
64	Quality of service not available
65	Error in SME (*)
70	Max submission attempt reached (finalised before validity period expired)
71	Max TTL for message reached (finalised when validity period expired)
72	Database sub-system error
73	Core dependency missing
74	Insufficient prepay credit
75	Core configuration Issue
76	Plug-in sub-system error
77	Routing loop detected
78	Age verification failure
79	Age verification failure
7A	Message in flight with unknown status
7B	Expenditure limit reached

(*) SME = Short Message Entity, an entity which may send or receive SMS.