RML Labroc HTTP Specifications

Version: 1.0.3

Date: February, 2018

Document Information and Version History

Document Information

Document Name	RML Labroc HTTP Specifications
Document Description	This document gives details on how to send messages via the Bulk HTTP API for the Route Mobile Labroc System
Creation Date	March 31, 2015
Initial Version	1.0.0
Author	Technical team

Document Change Log

Modification Date	Version Number	Change Summary	Author
January 24, 2016	1.0.1	Reformatted and reviewed document for language.	Technical team
February 08, 2017	1.0.2	Reformatted and reviewed document for language.	Technical team
February 14, 2018	1.0.3	Reformatted and reviewed document for language.	Hrudeep Goregaonkar

Document Approval Log

Approval Date	Approver	Designation
February 14, 2018	Sachin Kanojia	Head – Support

HTTP API to submit messages on SMPP

http://IP:port/bulksms/sendsms?username=XXXX&password=YYYYY&type=Y&dlr=Z&destination=QQQQQQQQQ&sou rce=RRRR&message=SSSSSSSS<&url=KKKK>

IP : smpp5.routesms.com

port: 8080 or 8000 (Use port 8000 if you are not able to connect using port 8080).

API parameters are explained below. Note that all the parameters, especially message and url should be URL-UTF-8 encoded.

Request Parameters		
Sr. No.	Parameter	Description
1	username	Username of the SMPP account.
2	password	Password of the SMPP account.
4	type	Indicates the type of the message. Values for "type" are: 0: Plain text (GSM 3.38 Character encoding) 1: Flash message (GSM 3.38 Character encoding) 2: Unicode 3: Reserved 4: WAP Push 5: Plain text (ISO-8859-1 Character encoding) 6: Unicode Flash 7: Flash Message (ISO-8859-1 Character encoding)
5	dlr	Indicates whether the client wants delivery report for this message. Range of values for "dlr" are: 0: No delivery report required 1: Delivery report required
7	destination	Mobile number to which to send the message (may or may not include a [+] sign). Multiple mobile numbers need to be separated by a comma [,] (the comma should be URL encoded).
	source	The source address in the message. Max length of 18 if only numeric. Max length of 11 if alphanumeric. To prefix the plus sign [+] to the sender's address when the message is displayed on their mobile phone, please prefix the plus sign to your sender's address while submitting the message (note the plus sign should be URL encoded). Additional restrictions on this field may be enforced by the SMSC.
8	message	The message to be sent. It can be used for 'long' messages, that

		is, messages longer than 160 characters for plain text, 140 for flash and 280 for Unicode. For concatenated (long) messages every 153 characters are counted as one message for plain text and 268 characters for Unicode, as the rest of the characters will be used by the system for packing extra information for re-assembling the message on the mobile phone. In case of WAP Push (type=4), this is the text that would appear in the message. Also in the latter case, to send non-English characters in the message, you need to directly URL encode them (using UTF-8 character encoding scheme).
9	url	If sending a WAP Push message (type=4), this parameter holds the link that you wish to send. For any other type of message, no value needs to be supplied for this parameter (if specified it will be ignored). This parameter should be URL encoded with UTF-8 character encoding (even for sending non-ASCII domain names).

Response		
Sr. No.	Error Code	Description
1	1701	Message sent successfully. You will receive the response 1701|<CELL_NO>|<MESSAGE ID>. The message id can then be used later to map the delivery reports to this message.
2	1702	Invalid URL Error. One of the parameters was not provided or left blank.
3	1703	Invalid value in username or password field.
4	1704	Invalid message type.
5	1705	Invalid message.
6	1706	Invalid destination.
7	1707	Invalid source (sender).
	1708	Invalid value for "dlr" field.
8	1709	User validation failed.
9	1710	Internal error.
12	1025	Insufficient user credit.

Note:

Along with the above errors codes, standard SMPP v3.4 error codes may also be returned where applicable.

Apart from 1709, Please **DO NOT RETRY** re-sending the message for any other error code (including SMPP v3.4 Error codes).

For better security you can also use the HTTPS protocol for submitting your messages so that your requests are encrypted when being sent over the Internet. To use the HTTPS protocol (when calling this link from a browser) just change the protocol in the link from HTTP to HTTPS and use port "8181" instead of "8080" as illustrated below (note that you might get a security error as Route Mobile currently uses a self-signed certificate. You will need to add an exception to continue with the message sending process):-
https://smpp5.routesms.com:8181/bulksms/sendsms?username=XXXX&password=YYYYY&type=Y&dlr=Z&destination=QQQQQQQ QQ&source=RRRR&message=SSSSSSSS

While sending messages with type=0 there are some special characters which end up taking the space of more than one character after final encoding and while sending to the SMSC. The following is the list of such known characters:

Character	Character	GSM 03.38
^	Circumflex Accent	0x1B14
{	Left Curly Bracket	0x1B28
}	Right Curly Bracket	0x1B29
\	Reverse Solidus (Backslash	0x1B2F
[	Left Square Bracket	0x1B3C
~	Tilde	0x1B3D
]	Right Square Bracket	0x1B3E
|	Vertical Bar	0x1B40
€	Euro Sign	0x1B65

Please do note that the above list is not exhaustive and hence before sending any special characters do test to see if they are supported.

Notes on WAP Push:

WAP Push messages are sent in binary mode and hence only 140 eight-bit characters (octets) are allowed in a single message. Also due to the nature of the message, approximately 32 characters are used to pack the document encoding standards so that the phone recognises the message correctly. As a result, only approximately 108 characters are available to send in each message (URL + message to display). However, if your URL contains some common patterns, they are encoded to occupy less space as shown below.

Pattern	Length	Actual Length
http://	7	1
http://www.	11	1
https://	8	1
https://www.	12	1
.com/ .edu/ .net/ .org/	4	1 [if at the end of the link, eg:http://www.routemobile.com/] 3 [if in the link, eg: http://www.routemobile.com/bulksms]

Apart from the exceptions in the previous table, all other characters are encoded using the UTF-8 character encoding, thus expanding the range of characters supported in a WAP Push (both "message" and "url"). Please refer to the home page of the Unicode Consortium for more details, like list of supported characters and how many octets a specified character would consume as per the UTF-8 encoding scheme.

Long WAP Push messages: If the total message length after encoding the document encoding standards with the message text and the URL exceed 140 octets, your account will automatically be charged the excess credits. For WAP Push messages longer than 140 octets, the number of octets usable for the message text and URL in each message part is further reduced from approximately 108 to 103 only.

Bulk SMS API Reply Format:

<Error_Code>|<destination>:<message_id>,<Error_Code>| <destination>:<message_id>...

Exceptional Situations:

- A request containing multiple destinations will be aborted immediately if any error other than "Invalid Destination" is found, in case an invalid destination is found we just skip that destination and proceed to the next destination.

- If while processing the request the SMPP Server goes down, the HTTP API will retry a fixed number (with a gap of ten milliseconds between consecutive retries) of times to reconnect to the SMPP server and submit the message. If the SMPP server does not come up before the fixed number of attempts are exhausted, the batch will be aborted at that destination and a message will be returned in following format:

  <Error_Code>|<destination>:<message_id>,<Error_Code>|<destination>:<message_id>,1709|<destination_at_which_batch_aborted>

- The credits can get exhausted in the middle of a request being serviced. In case such a situation occurs we will be aborting the batch on the destination at which we got the "Insufficient_Credit" error, and a response in the following format will be returned to the client:

  <Error_Code>|<destination>:<message_id>,<Error_Code>|<destination>:<message_id>,1025|<destination_at_which_batch_aborted>

Example Link to Submit Plain Text Messages (GSM 03.38 character set):

http://smpp5.routesms.com:8080/bulksms/sendsms?username=XXXX&password=YYYYY&type=0&dlr=1&destination= %2BXXXXXXXX&source=routesms&message=Demo%20Message!!!

The following observations can be made in the above URL:

- 'type=0', indicates this is a message of type plain text. This mode supports all characters falling under the GSM 03.38 character set.

- 'dlr=1', indicates delivery report for this message is enabled.
- 'message=Demo%20Message!!!', The message field contains the content to send in an URL encoded format. On using the appropriate username and password in the above link you will get the 'Demo Message!!!' on your mobile phone.
- 'destination=%2BXXXXXXX', An optional plus is included in the destination field here. Do note that the [+] sign is URL encoded.

Example Link to Submit Plain Text Messages (ISO-8859-1 Character set):

http://smpp5.routesms.com:8080/bulksms/sendsms?username=XXXX&password=YYYYY&type=5&dlr=0&destination=XXXXXXX&source=routesms&message=Demo%20Message!!!

The following observations can be made in the above URL:

- 'type=5', indicates message is of plain text type. This mode supports all characters falling under the ISO-8859-1 character set.
- 'dlr=0', indicates delivery report for this message is not enabled.
- 'message=Demo%20Message!!!', the message field contains the message to send in an URL encoded format, on using the appropriate username and password in the above link you will get the message "Demo Message!!!" on your mobile phone.
- 'destination=XXXXXXX' , the optional [+] has been omitted.

Example Link to Submit Flash Messages (GSM 03.38 Character set):

http://smpp5.routesms.com:8080/bulksms/sendsms?username=XXXX&password=YYYYY&type=1&dlr=0&destination=XXXXXXX&source=routesms&message=Demo%20Message!!!

On calling the above link by replacing the username and password by your account credentials, the message 'Demo Message!!!' should display on your mobile phone.

The characters in the message field should fall in the GSM 03.38 character set and the type parameter has to be set to 1 i. e. (type=1).

Example Link to Submit Unicode Flash Messages:

http://smpp5.routesms.com:8080/bulksms/sendsms?username=XXXX&password=YYYYY&type=6&dlr=0&destination=XXXXXXX&source=routesms&message=004400 65006D006F0020004D00650073007300610067006500 02100210021

On calling the above link by replacing the username and password by your account credentials, the message, 'Demo Message!!!' should display on the mobile phone.

The message needs to be encoded in the UTF-16BE format and the type parameter has to be set to 6 i.e. (type=6).

Example Link to Submit Unicode Messages

http://smpp5.routesms.com:8080/bulksms/sendsms?username=XXXX&password=YYYYY&type=2&dlr=0&destination=XXXXXXX&source=routesms&message=00440065006D006F0020004D006500730073006100670065 002100210021

On calling the above link by replacing the username and password by your account credentials, you should get the message, 'Demo Message!!!' should display on the mobile phone.

The message has to be encoded on the UTF-16BE format and the type parameter has to be set to 6 i.e. (type=6).

Example Link to Submit WAP Push Messages:

http://smpp5.routesms.com:8080/bulksms/sendsms?username=XXXX&password=YYYYY&type=4&dlr=1&destination=XXXXXXX&source=routesms&message=Test%20WAP%20Push%20Special%20Characters%3A%20%C2%B6%C2%BE%20Arabic%3A%20%D8%A6!!!&url=http%3A%2F%2Fwww.routesms.com%2F

The following observations can be made in the above URL:

- 'type=4', indicates this is a WAP Push, this mode supports all characters falling under the Unicode character set both in 'message' field and 'url'.
- 'dlr=1', indicates delivery report for this message is needed.
- 'message=Test%20WAP%20Push%20Special%20Characters%3A%20%C2%B6%C2%BE%20Arabic%3A%20%D8%A6%20Greek%3A%20%CD %BC!!!', The message field contained the message to displayed in URL encoded format, on using the appropriate username and password in the above link you will get a service message with text, 'Test WAP Push Special Characters: ¶¾ Arabic: ئ Greek: ϵ', on the specified mobile phone, hyper linked to the web address: 'http://www.routesms.com/'
- 'url=http%3A%2F%2Fwww.routesms.com%2F', URL Encoded format of 'http://www.routesms.com/'

Calling HTTP API Using .Net

```
Imports System.IO
Imports System.Net
Imports System.Data
Partial Class SendUsingSMPP
Inherits System.Web.UI.Page
Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
Dim WebRequest As Net.WebRequest 'object for WebRequest
Dim WebResonse As Net.WebResponse 'object for WebResponse
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''' ''''''''
' DEFINE PARAMETERS USED IN URL
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''' ''''''''
'To what server you need to connect to for submission
'i.e. Dim Server As String = "smpp1.routesms.com"
Dim Server As String = ""
'Port that is to be used
like 8080 or 8000 Dim
Port As String = ""
'Username that is to be used for submission
'i.e. Dim UserName As String = "tester" Dim UserName As String = ""
' password that is to be used along with username
'i.e. Dim Password As String = "password"
Dim Password As String = ""
'What type of the message that is to be sent.
'0:means plain text
'1:means flash
'2:means Unicode (Message content should be in Hex)
'6:means Unicode Flash(Message content should be in Hex) Dim type As Integer = 0
'Message content that is to be transmitted
Dim Message As String = "Test Message" 'Url Encode message
Message = HttpUtility.UrlEncode(Message)
If (Message = 2) Or (Message = 6) Then
       Message = ConvertToUnicode(Message)
End If
'Require DLR or not
'0:means DLR is not Required
'1:means DLR is Required Dim DLR As Integer = 1
'Sender Id to be used for submitting the message
'i.e. Dim SenderName As String = "test"
Dim Source As String = ""
'Destinations to which message is to be sent For submitting more
than one
'destination at once destinations should be comma separated Like
'XXXXXXX,XXXXXXX
Dim Destination As String = ""
'''''''CODE COMPLETE TO DEFINE PARAMETER'''''''''''''''
Dim WebResponseString As String = ""
Dim URL As String = "http://" & Server & ":" & Port &
"/bulksms/?username=" & UserName & "&password=" & Password & "&type=" &
type & "&dlr=" & DLR & "&destination=" & Destination & "&source="
&Source & "&message=" & Message & ""
WebRequest = Net.HttpWebRequest.Create(URL) 'Hit URL Link
WebRequest.Timeout = 25000
Try
```



```
WebResonse = WebRequest.GetResponse 'Get Response
Dim reader As IO.StreamReader = New
IO.StreamReader(WebResonse.GetResponseStream)
'Read Response and store in variable
WebResponseString = reader.ReadToEnd()
WebResonse.Close()
Response.Write(WebResponseString) 'Display Response.
Catch ex As Exception
WebResponseString = "Request Timeout"
'If any exception occur.
Response.Write(WebResponseString)
End Try
End Sub
'Function To Convert String to Unicode if MessageType=2 and 6.
Public Function ConvertToUnicode(ByVal str As String) As String
Dim ArrayOFBytes() As Byte =
System.Text.Encoding.Unicode.GetBytes(str)
Dim UnicodeString As String = ""
Dim v As Integer
For v = 0 To ArrayOFBytes.Length - 1
If v Mod 2 = 0 Then
Dim t As Integer = ArrayOFBytes(v)
ArrayOFBytes(v) = ArrayOFBytes(v + 1)
ArrayOFBytes(v + 1) = t
End If
Next
For v = 0 To ArrayOFBytes.Length - 1
Dim c As String = Hex$(ArrayOFBytes(v))
If c.Length = 1 Then
c = "0" & c
End If
UnicodeString = UnicodeString & c
Next
Return UnicodeString
End Function
End Class
```

Calling HTTP API Using PHP

```
<?php
class Sender{
var $host;
var $port;
/*
* Username that is to be used for submission
*/
var $strUserName;
/*
* password that is to be used along with username
*/
var $strPassword;
/*
* Sender Id to be used for submitting the message
*/
var $strSender;
```

```
/*
* Message content that is to be transmitted
*/
var $strMessage;
/*
* Mobile No is to be transmitted.
*/
var $strMobile;
/*
* What type of the message that is to be sent
* <ul>
* <li>0:means plain text</li>
* <li>1:means flash</li>
* <li>2:means Unicode (Message content should be in Hex)</li>
* <li>6:means Unicode Flash (Message content
should be in Hex)</li> * </ul>
*/
var $strMessageType;
/*
* Require DLR or not
* <ul>
* <li>0:means DLR is not Required</li>
* <li>1:means DLR is Required</li>
* </ul>
*/
var $strDlr;
private function sms__unicode($message){
$hex1='';
if (function_exists('iconv')) {
$latin = @iconv('UTF-8', 'ISO-8859-1', $message);
if (strcmp($latin, $message)) {
$arr = unpack('H*hex', @iconv('UTF-8', 'UCS-
2BE', $message));
$hex1 = strtoupper($arr['hex']);
}
if($hex1 ==''){
$hex2=''; $hex='';
for ($i=0; $i < strlen($message); $i++){
$hex = dechex(ord($message[$i]));  $len =strlen($hex);
$add = 4 - $len;
if($len < 4){
for($j=0;$j<$add;$j++){
$hex="0".$hex;
}
}
$hex2.=$hex;
}
return $hex2;
}
else{
return $hex1;
}
}else{
print 'iconv Function Not Exists !';
}
}  //Constructor..
```

```
public function Sender ($host,$port,$username,$password,$sender, $message,$mobile,
$msgtype,$dlr){
$this->host=$host;
$this->port=$port;
$this->strUserName = $username;
$this->strPassword = $password;
$this->strSender= $sender;
$this->strMessage=$message; //URL Encode The Message..
$this->strMobile=$mobile;
$this->strMessageType=$msgtype;
$this->strDlr=$dlr;
}
public function Submit(){
if($this->strMessageType=="2"||
$this->strMessageType=="6") {
//Call The Function Of String To HEX.
$this->strMessage =  $this->sms__unicode($this->strMessage);

try{
//Smpp http Url to send sms.
$live_url="http://".$this->host.":".$this-
>port."/sendsms?username=".$this-
>strUserName."&password=".$this->strPassword."&type=".$this-
>strMessageType."&dlr=".$this->strDlr."&destination=".$this-
>strMobile."&source=".$this->strSender."&message=".$this-
>strMessage."";
$parse_url=file($live_url);
echo $parse_url[0];
}catch(Exception $e){
echo 'Message:' .$e->getMessage();
}

}
else
$this->strMessage=urlencode($this->strMessage);
try{
//Smpp http Url to send sms.
$live_url="http://".$this->host.":".
$this->port."/sendsms?username=".$this->strUserName."&password=".$this-
>strPassword."&type=".$this->strMessageType."&dlr=".$this-
>strDlr."&destination=".$this-
>strMobile."&source=".$this-
>strSender."&message=".$this->strMessage."";
$parse_url=file($live_url);
echo $parse_url[0];
}
catch(Exception $e){
echo 'Message:' .$e->getMessage();
}
}
}
//Call The Constructor.
$obj = new Sender("IP","Port","","","Tester"," "919990001245 ","ةيبرعلا
","2","1");
$obj->Submit ();
?>
```

Calling HTTP API Using Java

```
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
/**
* An Example Class to use for the submission using HTTP API You can perform
* your own validations into this Class For username, password, destination,
* source, dlr, type, message, server and port
**/
public class Sender {
// Username that is to be used for submission
String username;
// password that is to be used along with username
String password;
// Message content that is to be transmitted
String message;
/**
* What type of the message that is to be sent
* <ul>
* <li>0:means plain text</li>
* <li>1:means flash</li>
* <li>2:means Unicode (Message content should be in Hex)</li>
* <li>6:means Unicode Flash (Message content should be in Hex)</li>
* </ul>
*/
String type;
/**
* Require DLR or not
* <ul>
* <li>0:means DLR is not Required</li>
* <li>1:means DLR is Required</li>
* </ul>
*/
String dlr;
/**
* Destinations to which message is to be sent For submitting more than one
* destination at once destinations should be comma separated Like
* 91999000123,91999000124
*/
String destination;
// Sender Id to be used for submitting the message
String source;
// To what server you need to connect to for submission
String server;
// Port that is to be used like 8080 or 8000
int port;
public Sender(String server, int port, String username, String password,
String message, String dlr, String type, String destination,
String source) {
```

```
this.username = username; this.password = password; this.message = message; this.dlr =
dlr;
this.type = type;
this.destination = destination; this.source = source; this.server = server;
this.port = port;
}
private void submitMessage() {
try {
      // Url that will be called to submit the message
      URL sendUrl = new URL("http://" + this.server + ":" + this.port
      + "/bulksms/bulksms");
      HttpURLConnection httpConnection = (HttpURLConnection) sendUrl
      .openConnection();
      // This method sets the method type to POST so that
      // will be send as a POST request
      httpConnection.setRequestMethod("POST");
      // This method is set as true wince we intend to send
      // input to the server
      httpConnection.setDoInput(true);
      // This method implies that we intend to receive data from server.
      httpConnection.setDoOutput(true);
      // Implies do not use cached data
      httpConnection.setUseCaches(false);
      // Data that will be sent over the stream to the server.
      DataOutputStream dataStreamToServer = new DataOutputStream(
      httpConnection.getOutputStream());
      dataStreamToServer.writeBytes("username="
      + URLEncoder.encode(this.username, "UTF-8") + "&password="
      + URLEncoder.encode(this.password, "UTF-8") + "&type="
      + URLEncoder.encode(this.type, "UTF-8") + "&dlr="
      + URLEncoder.encode(this.dlr, "UTF-8") + "&destination="
      + URLEncoder.encode(this.destination, "UTF-8") + "&source="
      + URLEncoder.encode(this.source, "UTF-8") + "&message="
      + URLEncoder.encode(this.message, "UTF-8"));
      dataStreamToServer.flush();
      dataStreamToServer.close();
      // Here take the output value of the server.
      BufferedReader dataStreamFromUrl = new BufferedReader(
      new InputStreamReader(httpConnection.getInputStream()));
      String dataFromUrl = "", dataBuffer = "";
      // Writing information from the stream to the buffer

      while ((dataBuffer = dataStreamFromUrl.readLine()) != null) {
            dataFromUrl += dataBuffer;
             }
      /**
      * Now dataFromUrl variable contains the Response received from the
      * server so we can parse the response and process it accordingly.
      */
      dataStreamFromUrl.close();
      System.out.println("Response: " + dataFromUrl);
} catch (Exception ex) {
      ex.printStackTrace();
}
public static void main(String[] args) {
try {
```

```
        // Below exmaple is for sending Plain text
        Sender s = new Sender("smpp2.routesms.com", 8080, "tester909",
        "test11", "test for unicode", "1", "0", "919869533416",
        "Update");
        s.submitMessage();
        // Below exmaple is for sending unicode
        Sender s1 = new Sender("smpp2.routesms.com", 8080, "xxxx",
        "xxx", convertToUnicode("test for unicode").toString(),
        "1", "2", "919869533416", "Update");
        s1.submitMessage();
} catch (Exception ex) {
}
}

/**
* Below method converts the unicode to hex value
* @param regText
* @return
*/
private static StringBuffer convertToUnicode(String regText) {
char[] chars = regText.toCharArray();
        StringBuffer hexString = new StringBuffer();
        For (int i = 0; i < chars.length; i++) {
        String iniHexString = Integer.toHexString((int) chars[i]);
        If (iniHexString.length() == 1)
              iniHexString = "000" + iniHexString;
              else if (iniHexString.length() == 2)
              iniHexString = "00" + iniHexString;
              else if (iniHexString.length() == 3)
                        iniHexString = "0" + iniHexString;
             hexString.append(iniHexString
         );
}
System.out.println(hexString);
return hexString;
}
}
```

Appendix

GSM 03.38 Character set

GSM 03.38																
	x0	**x1**	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**x8**	**x9**	**xA**	**xB**	**xC**	**xD**	**xE**	**xF**
0x	@	£	$	¥	è	é	ù	ì	ò	Ç	LF	Ø	ø	CR	Å	å
1x	Δ	_	Φ	Γ	Λ	Ω	Π	Ψ	Σ	Θ	Ξ	ESC	Æ	æ	ß	É
2x	SP	!	"	#	¤	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	¡	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	Ä	Ö	Ñ	Ü	§
6x	¿	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	ä	ö	ñ	ü	à
1B 0x												FF				
1B 1x					^											
1B 2x									{	}						\
1B 3x													[	~	]	
1B 4x	|														
1B 5x																
1B 6x						€										
1B 7x																

ISO-8859-1 Character set

ISO/IEC 8859-1															
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	unused															
1x																
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-:	.	/
3x	0	1	2	3	4	5	6	7	8	9	;	<	= J		>	?
4x	@	A	B	C	D	E	F	G	H	I	K	L	M		N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8x	unused															
9x																
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Code values 00–1F, 7F, and 80–9F are not assigned to characters by ISO/IEC 8859-1.