Route Mobile

SMSPLUS – Bulk HTTP API Specifications

Version: 1.0.1
Date: January, 2018
# Document Information and Version History

## Document Information

<table>
<thead>
<tr>
<th>Document Name</th>
<th>Route Mobile – SMSPLUS – Bulk HTTP API Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document Description</td>
<td>This document details sending messages via the bulk HTTP API for the system</td>
</tr>
<tr>
<td>Creation Date</td>
<td>April 07, 2017</td>
</tr>
<tr>
<td>Initial Version</td>
<td>1.0.0</td>
</tr>
<tr>
<td>Author</td>
<td>Priyanka</td>
</tr>
</tbody>
</table>

## Document Change Log

<table>
<thead>
<tr>
<th>Modification Date</th>
<th>Version Number</th>
<th>Change Summary</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 24, 2018</td>
<td>1.0.1</td>
<td>Restructured and reformatted document. Replaced mobile phone no. in ‘Destination’ field with ‘xxxxxxx’. Added error codes under section 'HTTP API to Submit Messages on SMPP'.</td>
<td>Hrudeep Goregaonkar</td>
</tr>
</tbody>
</table>

## Document Approval Log

<table>
<thead>
<tr>
<th>Approval Date</th>
<th>Approver</th>
<th>Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 24, 2018</td>
<td>Sachin Kanojia</td>
<td>Head – Support</td>
</tr>
</tbody>
</table>
HTTP API to Submit Messages on SMPP

http://<server>:8080/bulksms/bulksms?
username=XXXX&password=YYYYY&type=Y&dlr=Z&destination=QQQQQQQQQ&source=RRRRR&message=SSSSSSSSS&lt;url=KKK&gt;

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

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>username</td>
<td>The username of the HTTP account.</td>
</tr>
<tr>
<td>2</td>
<td>password</td>
<td>The password of the HTTP account.</td>
</tr>
<tr>
<td>3</td>
<td>type</td>
<td>It indicates the type of message. Values for type include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Plain text (GSM 3.38 Character encoding)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Flash (GSM 3.38 Character encoding)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: Unicode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4: WAP Push</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5: Plain text (ISO-8859-1 Character encoding)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6: Unicode Flash</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7: Flash (ISO-8859-1 Character encoding)</td>
</tr>
<tr>
<td>4</td>
<td>dlr</td>
<td>Indicates whether the client wants delivery report for this message. The values for dlr include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: No delivery report required</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Delivery report required</td>
</tr>
<tr>
<td>5</td>
<td>destination</td>
<td>Mobile number to which the message is to be sent (may or may not include a plus [+ ] sign). Multiple mobile numbers can be separated by commas (,) (the comma should be URL encoded).</td>
</tr>
<tr>
<td>6</td>
<td>source</td>
<td>The source address that should appear in the message.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Max Length of 18 if numeric.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Max Length of 11 if alphanumeric.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To prefix the plus sign (+) to the sender's address when the message is displayed on their cell 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.</td>
</tr>
</tbody>
</table>
| 7       | 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
Request Parameters

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>url</td>
<td>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 will be ignored). This parameter should be URL encoded with UTF-8 character encoding (even for sending non-ASCII domain names).</td>
</tr>
</tbody>
</table>

Error Codes

Error codes are explained in the following table.

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1701</td>
<td>Success, Message Submitted Successfully. In this case you will receive the response 1701</td>
</tr>
<tr>
<td>2</td>
<td>1702</td>
<td>Invalid URL. This means that one of the parameters was not provided or left blank.</td>
</tr>
<tr>
<td>3</td>
<td>1703</td>
<td>Invalid value in username or password parameter.</td>
</tr>
<tr>
<td>4</td>
<td>1704</td>
<td>Invalid value in type parameter.</td>
</tr>
<tr>
<td>5</td>
<td>1705</td>
<td>Invalid message.</td>
</tr>
<tr>
<td>6</td>
<td>1706</td>
<td>Invalid destination.</td>
</tr>
<tr>
<td>7</td>
<td>1707</td>
<td>Invalid source (Sender).</td>
</tr>
<tr>
<td>8</td>
<td>1708</td>
<td>Invalid value for dlr parameter.</td>
</tr>
<tr>
<td>9</td>
<td>1709</td>
<td>User validation failed.</td>
</tr>
<tr>
<td>10</td>
<td>1710</td>
<td>Internal error.</td>
</tr>
<tr>
<td>11</td>
<td>1025</td>
<td>Insufficient credit.</td>
</tr>
<tr>
<td>12</td>
<td>1715</td>
<td>Response timeout.</td>
</tr>
<tr>
<td>13</td>
<td>1032</td>
<td>DND reject.</td>
</tr>
<tr>
<td>14</td>
<td>1028</td>
<td>Spam message.</td>
</tr>
</tbody>
</table>
Note:

- Along with the above errors codes, standard SMPP v3.4 error codes may also be returned where applicable.
- Apart from error code 1709, please do NOT retry re-sending the message for any other error code (including SMPP v3.4 Error codes).
- For error code 1715, do NOT re-submit the same message again.

**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. In case 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)**


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://<server>:8080/bulksms/bulksms?
username=XXXX&password=YYYY&type=5&dlr=0&destination=xxxxxxx&source=xxxxxxx&message=Demo%20Message!!!

The following observations can be made in the above URL:

• 'type=5', indicates message is of type plain text. 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://<server>:8080/bulksms/bulksms?
username=XXXX&password=YYYY&type=1&dlr=0&destination=xxxxxxx&source=xxxxxxx&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 cell 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://<server>:8080/bulksms/bulksms?
username=XXXX&password=YYYY&type=6&dlr=0&destination=xxxxxxx&source
On calling the above link by replacing the username and password by your account credentials, the sms 'Demo Message!!!' should flash on the mobile number in the destination field.

The message has 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://<server>:8080/bulksms/bulksms?
username=XXXX&password=YYYY&type=2&dlr=0&destination=xxxxxxx&source=xxxxxxx&message=0044065005006D006F0020004D006500730073006100670065002100210021

On calling the above link by replacing the username and password by your account credentials, you should get the SMS ‘Demo Message!!!’ on the mobile number in the destination field.

The message has to be encoded on the UTF-16BE format and the type parameter has to be set to 6 i.e. (type=2).

**Calling HTTP API using .Net**

```csharp
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 WebResponse 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 = "xxxxx.xxxx.xxxx"
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/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
WebResponse = WebRequest.GetResponse 'Get Response
Dim reader As IO.StreamReader = New
IO.StreamReader(WebResponse.GetResponseStream) 'Read
Response and store in variable WebResponseString = reader.ReadToEnd() WebResponse.Close()
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 Message Type = 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)
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
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', $message);
    if (strncmp($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 $hex1;
} }
} //Constructor..
public function Sender ($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."/".$this->port."/".$this->strUserName."&username=".$this->strPassword."&message=".$this->strMessageType."&message=".$this->strSender."&destination=".$this->strMobile."&source=".$this->strMessage."";
            $parse_url=file($live_url);
            echo $parse_url[0];
            }catch(Exception $e){
                echo 'Message:' . $e->getMessage();
            }
    }
    $this->strMessage=urlencode($this->strMessage);
    try{
        // http Url to send sms.

        $live_url="http://".$this->host":".$this->port."/".$this->strUserName."&username=".$this->strPassword."&message=".$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 {
        print 'iconv Function Not Exists !';
    }
}
catch(Exception $e){
    echo 'Message:' . $e->getMessage();
}

//Call The Constructor.
$obj = new Sender("IP","Port","","","Tester"," " xxxxxxxx ","2","1");
$obj->Submit();
?

### Calling HTTP API using Java

```java
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLDecoder;
/**
 * 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
* xxxxxxx,xxxxxxx */
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 which 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());
  }
  catch (Exception e) {
  }
}
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 example is for sending Plain text
    Sender s = new Sender("smpp2.xxxxxxxx.com", 8080, "tester909",
      "test1", "test for unicode", "1", "0", "xxxxxxxx",
      "Update");
    s.submitMessage();
    // Below example is for sending unicode
    Sender s1 = new Sender("smpp2.xxxxxxxx.com", 8080, "xxxx",
      "xxx", convertToUnicode("test for unicode").toString(),
      "1", "2", "xxxxxxxx", "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  | ö  | Ç  | LF | Ø  | ø  | CR | Å  | â  |
| 1x        | Δ  | _  | Ø  | 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        | i  | 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  | Ù  | Å  | Ì  |
| 6x        | á  | a  | b  | c  | d  | e  | f  | g  | h  | i  | j  | k  | l  | m  | n  | o  |
| 7x        | ï  | ð  | ð  | ð  | ð  | ð  | ð  | ð  | ð  | ð  | ð  | ð  | ð  | ð  | ð  |
| 1B 0x      | FF |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 1B 1x      | ^  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 1B 2x      | {  | }  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 1B 3x      | [  | ~  | ]  |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 1B 4x      | |  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 1B 5x      | |  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 1B 6x      | €  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 1B 7x      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
# ISO-8859-1 Character set

<table>
<thead>
<tr>
<th>ISO/IEC 8859-1</th>
<th>x0</th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
<th>x9</th>
<th>xA</th>
<th>xB</th>
<th>xC</th>
<th>xD</th>
<th>xE</th>
<th>xF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1x</td>
<td>unused</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2x</td>
<td>SP</td>
<td>!</td>
<td>&quot;</td>
<td>#</td>
<td>$</td>
<td>%</td>
<td>&amp;</td>
<td>'</td>
<td>(</td>
<td>)</td>
<td>*</td>
<td>+</td>
<td>,</td>
<td>-</td>
<td>.</td>
<td>/</td>
</tr>
<tr>
<td>3x</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>:</td>
<td>;</td>
<td>&lt;</td>
<td>=</td>
<td>&gt;</td>
<td>?</td>
</tr>
<tr>
<td>4x</td>
<td>@</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
<td>N</td>
<td>O</td>
</tr>
<tr>
<td>5x</td>
<td>P</td>
<td>Q</td>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>[</td>
<td>\</td>
<td>]</td>
<td>^</td>
<td>_</td>
</tr>
<tr>
<td>6x</td>
<td>`</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>o</td>
</tr>
<tr>
<td>7x</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>u</td>
<td>v</td>
<td>w</td>
<td>x</td>
<td>y</td>
<td>z</td>
<td>{</td>
<td></td>
<td></td>
<td>}</td>
<td>~</td>
</tr>
<tr>
<td>8x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9x</td>
<td>unused</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ax</td>
<td>NBS</td>
<td>P</td>
<td>c</td>
<td>£</td>
<td>¥</td>
<td>5</td>
<td>&quot;</td>
<td>©</td>
<td>a</td>
<td>«</td>
<td>S</td>
<td>Y</td>
<td>¥</td>
<td>©</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Bx</td>
<td>°</td>
<td>±</td>
<td>2</td>
<td>3</td>
<td>′</td>
<td>µ</td>
<td>1</td>
<td>0</td>
<td>»</td>
<td>¼</td>
<td>½</td>
<td>¾</td>
<td>⁻</td>
<td>ä</td>
<td>Æ</td>
<td></td>
</tr>
<tr>
<td>Cx</td>
<td>À</td>
<td>Á</td>
<td>Â</td>
<td>Ä</td>
<td>Å</td>
<td>Å</td>
<td>È</td>
<td>Ë</td>
<td>É</td>
<td>È</td>
<td>Ê</td>
<td>Ë</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
</tr>
<tr>
<td>Dx</td>
<td>Đ</td>
<td>Ñ</td>
<td>Õ</td>
<td>Ó</td>
<td>Ö</td>
<td>Ö</td>
<td>×</td>
<td>Õ</td>
<td>Õ</td>
<td>Ü</td>
<td>Ü</td>
<td>Ü</td>
<td>Ü</td>
<td>Ü</td>
<td>Ü</td>
<td>Ü</td>
</tr>
<tr>
<td>Ex</td>
<td>à</td>
<td>á</td>
<td>â</td>
<td>å</td>
<td>æ</td>
<td>æ</td>
<td>ç</td>
<td>è</td>
<td>è</td>
<td>è</td>
<td>è</td>
<td>è</td>
<td>è</td>
<td>è</td>
<td>è</td>
<td>è</td>
</tr>
<tr>
<td>Fx</td>
<td>ð</td>
<td>ñ</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
</tr>
</tbody>
</table>

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