



## Software Development Kit & Short Code Guide

---

### **Abstract**

This document explains how to use short codes for both premium and non-premium SMS using the OpenMarket™ Software Development Kit.

## Revision History

Date	Version	Description
6/2/2004	1.0.0	- First Draft
10/10/2004	1.0.1	- Added example code for each SDK that supports short codes. - Added process flow for short code provisioning and testing.
12/4/2004	1.0.2	- Added tips throughout document that are highlighted on the left side throughout document.
1/14/2005	1.0.3	- Minor corrections made throughout document
3/2/2005	1.0.4	- Added PHP SDK supporting information and example code. - Added more information on what a web server should return on an inbound MO SMS - Added error checking for "deliver" type on any inbound/posted XML - Added documentation on how to extract which carrier/operator an inbound MO SMS originated from
6/15/2005	1.0.5	- Changed formatting of document. - Added Perl SMS SDK supporting information and example code. - Fixed PHP deliver example code. - Removed "Quick Start" section and merged into "Sending SMS" section.

---

## Table of Contents

<b>Introduction.....</b>	<b>2</b>
Check List	2
Software Development Kit Version	3
<b>Sending SMS .....</b>	<b>4</b>
ActiveX SMS SDK & Visual Basic Script	4
Java SMS SDK	4
Perl SMS SDK	5
PHP SMS SDK	5
<b>Premium Billing.....</b>	<b>7</b>
Charge Type	7
ActiveX SMS SDK	7
Java SMS SDK	7
Perl SMS SDK	8
PHP SMS SDK	8
Charge Amount	8
ActiveX SMS SDK	8
Java SMS SDK	9
Perl SMS SDK	9
PHP SMS SDK	9
Premium Billing Example Code	9
ActiveX SMS SDK	9
Java SMS SDK	9
Perl SMS SDK	10
PHP SMS SDK	10
<b>Receiving SMS.....</b>	<b>12</b>
XML Protocol Version	12
Carrier Identification	12
HTTP Delivery	12
ActiveX SMS SDK, Microsoft IIS, Active Server Pages	13
Java SMS SDK, Apache, Java Server Pages	13
Perl SMS SDK, Apache, CGI Script	14
PHP SMS SDK, Apache, PHP Script	14
<b>Support .....</b>	<b>16</b>

---

## Introduction

A short code is a 3-6 digit phone number that is used for sending and receiving premium or non-premium SMS. Most customers are familiar with a full phone number, known as a long code, that is typically 7 or more digits in length depending on the numbering plan of the home country. For example, in North America, a long code is 11 digits in length (the first digit is the country code followed by a national number of 10 digits). In order to send and receive premium SMS or receive high volumes of non-premium SMS, network operators require the activation of a short code.

A short code is usually 5 digits in length, such as 55424, and is activated on a specific wireless network. The same 5 digit code can be activated with more than one operator, allowing user's to be able to interact with your application across carriers.

OpenMarket provides customers with the ability to use any one of its SMS Software Development Kits to take full advantage of short codes to send and receive SMS. Please note that every short code requires activation with OpenMarket. Otherwise, the OpenMarket Network will deny access.

You can contact our sales department at any time, by sending us a request from the web at <http://www.OpenMarket.com/contact/sales/>.

### Check List

The following tasks must be completed in order to successfully send and receive SMS using short codes with OpenMarket:

1. **Short Code Registration** – Go through the application process including the documentation of the short code program form, going through the OpenMarket approval process, and receiving carrier approvals.
2. **Short Code Activation** – Get one or more carriers to “turn on” your short code. Short codes are initially activated in test mode. Test mode allows non-production sending and receiving of SMS on the short code. Premium billing may be triggered as well, however, the actual billing will not be applied to a cell phone bill until the code is switched into production mode.
3. **Verify SMS Software Development Kit Version** – Ensure the kit you are using supports sending and receiving SMS with short codes (explained below).
4. **Verify OpenMarket XML Delivery Version** – Ensure OpenMarket is using version 3.0 of its XML format to forward

---

received SMS to your web server. OpenMarket defaults to version 2.0 for backwards compatibility on older SDKs. Only version 3.0 supports short codes.

5. **Implementation** - Implement the application that will handle the sending and receiving of SMS on your short code.
6. **Production Launch** - Once the application has been fully tested and approved by OpenMarket and the carrier(s), the short code will be placed into production mode. Once in production mode, premium billing charges will be live.

**Tip**

To find out which version of an SMS SDK is installed, please access the "UserAgent" property and print out its value.

For example:

**ActiveX SMS SDK**

*sms.UserAgent*

**Java SMS SDK**

*sms.getUserAgent()*

**Perl SMS SDK**

*\$sms->userAgent()*

**PHP SMS SDK**

*\$sms->getUserAgent()*

### Software Development Kit Version

The ability to use a short code with the OpenMarket SMS SDK will require a minimum version of each kit. The following SMS SDKs support short codes for sending and receiving premium and non-premium SMS:

- ActiveX SMS SDK – version 2.9.3 or higher
- Java SMS SDK – version 2.8.0 or higher
- Perl SMS SDK – version 2.7.0 or higher
- C/C++ SMS SDK – **not available** (customer responsible for XML parsing)
- PHP SMS SDK – version 4.0.0 Beta 1 or higher. Please note that version 4.0.0 is not source compatible with older versions of the PHP SMS SDK.
- Please see "Wireless Message Protocol Specifications 3.0 Manual" for more information about using OpenMarket's raw XML protocol for implementing your own SMS library.

---

## Sending SMS

### Tip

OpenMarket strictly enforces short code use. Remember that short codes are ONLY activated per specified carrier.

If you activated a short code of 10454 on T-Mobile in the United States, and you try to send to a Verizon phone using the short code of 10454, the OpenMarket Network will reject your submission.

Once you have verified your SMS SDK version supports short codes, your application should be ready to send SMS. It is also assumed that your short code has been activated on the OpenMarket Network and you have been notified of the activation by OpenMarket Support. The key to sending SMS using short codes is properly setting the source and destination address. Starting with recent versions of OpenMarket's SMS Software Development Kit, different types of addresses can be set. Address types include Unknown, International, Network/Short Code, and Alphanumeric. For purposes of this document, only the Network/Short Code and Unknown types of addresses will be explained.

For example, OpenMarket notifies you that the short code 55432 has been activated for your account. An SMS using that short code would be sent with the following code (only code specific to using short codes is shown, please refer to the samples directory provided with each SDK for more information).

### ActiveX SMS SDK & Visual Basic Script

Please see "send-text.asp" in the examples directory of the ActiveX SDK to see a full example of code for sending a non-premium SMS message with a short code.

```
<preceding code...>
' Requires ActiveX SMS SDK version 2.9.3 or higher
' Sets type to NETWORK (short code)
sms.SourceType = 3
' Specify which short code
sms.SourceAddr = "55432"
' Sets type to UNKNOWN (default)
sms.DestinationType = 0
' The cell phone of recipient
sms.DestinationAddr = "+13135551212"
' The text of the SMS message. Typically max 160 chars.
sms.MsgText = "Hello World From OpenMarket!"
<remaining code...>
```

### Java SMS SDK

Please see "send-text.java" in the examples directory of the Java SDK to see a full example of code for sending a non-premium SMS message with a short code.

```

<preceding code...>

// Requires Java SMS SDK version 2.8.1 or higher

// Sets type to NETWORK (short code) and sets short code
sms.setSourceAddr(SMS.ADDR_TYPE_NETWORK, "55432");

// Sets type to UNKNOWN (default) and sets phone # of recipient
sms.setDestinationAddr(SMS.ADDR_TYPE_UNKNOWN, "+13135551212");

// The text of the SMS message. Typically max 160 chars.
sms.setMsgText("Hello World From OpenMarket!");

<remaining code...>

```

## Perl SMS SDK

Please see "send\_text.pl" in the examples directory of the Perl SDK to see a full example of code for sending a non-premium SMS message with a short code.

```

<preceding code...>

# Requires Perl SMS SDK version 2.7.0 or higher

# Sets type to NETWORK (short code)
$sms->sourceType(ADDR_TYPE_NETWORK);

# Sets short code
$sms->sourceAddr("55432");

# Sets type to UNKNOWN (default)
$sms->destinationType(ADDR_TYPE_UNKNOWN);

# Sets cell phone # of recipient
$sms->destinationAddr("+13135551212");

# The text of the SMS message. Typically max 160 chars.
$sms->msgText("Hello World From OpenMarket!");

<remaining code...>

```

## PHP SMS SDK

Please see "submit-text.php" in the samples directory of the PHP SDK to see a full example of code for sending a non-premium SMS message with a short code.

```

<preceding code...>

// Requires PHP SMS SDK version 4.0.0 or higher

// Sets type to NETWORK (short code) and sets short code
$sms->setSourceAddr(new Address("55432", SMS_ADDR_TYPE_NETWORK));

// Sets type to UNKNOWN (default) and sets cell phone # of recipient

```

---

```
$sms->setDestinationAddr(new Address("+13135551212"));  
// The text of the SMS message. Typically max 160 chars.  
$sms->setMessageText("Hello World From OpenMarket!");  
  
<remaining code...>
```

## Premium Billing

Short codes can be configured to support premium billing, or the ability to charge the recipient a fee for your service. Premium messaging fees may be triggered on either mobile terminated (outgoing SMS to phone) or mobile originated (incoming SMS from phone) SMS.

Charging for premium fees on mobile terminated SMS offers the most flexibility and is the preferred method. Some countries and carriers do not support mobile terminated premium billing.

There are two optional parameters available in each SMS SDK to either trigger premium charges on mobile terminated or to extract out if charges occurred during a mobile originated SMS.

1. Charge Type
2. Charge Amount

### Note

Concatenated SMS such as ringtones or long text-messages should only set the first segment to a ChargeType of 1. If all segments are set to 1, then it will result in a charge for each segment.

Please contact OpenMarket Support if you would like your application verified regarding this scenario.

### Charge Type

The Charge Type optional property controls whether an SMS should trigger a charge to be placed on the recipient's mobile phone bill. This optional property allows the same short code to be used for both premium and non-premium SMS.

If this property is omitted from the submit request, OpenMarket will default its value to the account's pre-configured setting. Please note that while this typically defaults to 0 (non-premium), it is possible that the default is 1 (premium).

A Charge Type may be set to the following values:

- 0 = do not charge (non-premium SMS)
- 1 = charge recipient (premium SMS)

### ActiveX SMS SDK

```
sms.ChargeType = 0      ' Do not charge
sms.ChargeType = 1      ' Trigger charge
```

### Java SMS SDK

```
sms.setChargeType(0); // Do not charge
sms.setChargeType(1); // Trigger charge
```

## Perl SMS SDK

```
$sms->chargeType(0); # Do not charge  
$sms->chargeType(1); # Trigger charge
```

## PHP SMS SDK

```
$sms->setChargeType(0); // Do not charge  
$sms->setChargeType(1); // Trigger charge
```

## Charge Amount

The Charge Amount optional property controls which price tier will be used for the charge on the recipient's mobile phone bill. This optional property is only valid when all the following apply:

- The Charge Type property is set to 1
- The Charge Amount is pre-approved by OpenMarket.

The Charge Amount is an integer representation of the total number cents, pence, etc. to charge for the premium SMS. The currency of the amount is always considered local to the country of the destination address. For example, if the destination is a US recipient, the Charge Amount will be interpreted as US dollars. On the other hand, if the destination address is within Australia, then the amount will be interpreted as Australian dollars.

In the United States, for a \$1.99 charge, this value would be set to 199. In other countries such as the United Kingdom, for a £0.25 charge, this value would be set to 25.

A Charge Amount may be set to an integer value representing the amount to charge.

- \$1.99 charge should be 199 for this value.
- \$0.30 charge should be 30 for this value.
- \$1.50 charge should be 150 for this value.
- £0.25 charge should be 25 for this value.

## ActiveX SMS SDK

```
sms.ChargeAmount = 50 ' Charge $0.50 in USD
```

### Note

If the ChargeType property is set to 1, but the ChargeAmount property is omitted, OpenMarket will default to the customer's pre-configured default price tier.

---

### Java SMS SDK

```
sms.setChargeAmount(50); // Charge $0.50 in USD
```

### Perl SMS SDK

```
$sms->chargeAmount(50); # Charge $0.50 in USD
```

### PHP SMS SDK

```
$sms->setChargeAmount(50); // Charge $0.50 in USD
```

Premium Billing Example Code

### ActiveX SMS SDK

Please see "send-text.asp" in the examples directory of the ActiveX SDK to see a full example of code for sending a premium SMS message with a short code.

```
<preceding code...>  
' Sets type to NETWORK (short code)  
sms.SourceType = 3  
  
' Specify which short code  
sms.SourceAddr = "55432"  
  
' Sets type to UNKNOWN (default)  
sms.DestinationType = 0  
  
' The cell phone of recipient in international dialing format  
sms.DestinationAddr = "+13135551212"  
  
' The text of the SMS message. Typically max 160 chars.  
sms.MsgText = "You were just sent a premium SMS @ $1.99!"  
  
' Set premium rate to be charged  
sms.ChargeType = 1 ' Tell OpenMarket to bill on this SMS  
sms.ChargeAmount = 199 ' Set premium rate to $1.99 USD  
  
<remaining code...>
```

### Java SMS SDK

Please see "send-text.java" in the examples directory of the Java SDK to see a full example of code for sending a premium SMS message with a short code.

```

<preceding code...>

// Sets type to NETWORK (short code) and sets short code
sms.setSourceAddr(SMS.ADDR_TYPE_NETWORK, "55432");

// Sets type to UNKNOWN (default) and sets cell phone # of
recipient
sms.setDestinationAddr(SMS.ADDR_TYPE_UNKNOWN, "+13135551212");

// The text of the SMS message. Typically max 160 chars.
sms.setMsgText("You were just sent a premium SMS @ $1.99!");

// Set premium rate to be charged
sms.setChargeType(1); // Tell OpenMarket to bill on this SMS
sms.setChargeAmount(199); // Set premium rate to $1.99 USD

<remaining code...>

```

## Perl SMS SDK

Please see "send\_text.pl" in the examples directory of the Perl SDK to see a full example of code for sending a premium SMS message with a short code.

```

<preceding code...>

# Requires Perl SMS SDK version 2.7.0 or higher

# Sets type to NETWORK (short code)
$sms->sourceType(ADDR_TYPE_NETWORK);

# Sets short code
$sms->sourceAddr("55432");

# Sets type to UNKNOWN (default)
$sms->destinationType(ADDR_TYPE_UNKNOWN);

# Sets cell phone # of recipient
$sms->destinationAddr("+13135551212");

# The text of the SMS message. Typically max 160 chars.
$sms->msgText("Hello World From OpenMarket!");

# Set premium rate to be charged
$sms->chargeType(1); # Bill on this SMS
$sms->chargeAmount(199); # Set premium rate to $1.99 USD

<remaining code...>

```

## PHP SMS SDK

Please see "submit-text.php" in the samples directory of the PHP SDK to see a full example of code for sending a premium SMS message with a short code.

```

<preceding code...>

```

---

```
// Sets type to NETWORK (short code) and sets short code
$sms->setSourceAddr(new Address("55432", SMS_ADDR_TYPE_NETWORK));

// Sets type to UNKNOWN (default) and sets phone # of recipient
$sms->setDestinationAddr(new Address("+13135551212"));

// The text of the SMS message. Typically max 160 chars.
$sms->setMessageText("You were sent a premium SMS @ $1.99!");

// Set premium rate to be charged
$sms->setChargeType(1); // Bill on this SMS
$sms->setChargeAmount(199); // Set premium rate to $1.99 USD

<remaining code...>
```

---

## Receiving SMS

### XML Protocol Version

The SMS Software Development Kit uses an underlying XML-based protocol for either sending SMS or receiving SMS. There are currently two active versions of OpenMarket's XML protocol: version 2.0 and version 3.0. Only version 3.0 of OpenMarket's XML protocol supports short codes.

Before your account is configured to receive version 3.0, please ensure you are using a qualified SMS SDK to parse incoming XML packets. If you are using an older SDK that only supports version 2.0, your application will error out while parsing version 3.0 XML.

### Carrier Identification

On SMS sent to short codes from mobile handsets, OpenMarket will identify which carrier this occurred on and include this information in the XML sent to your system. OpenMarket has assigned a unique numeric code for each network operator. For example, T-Mobile in the United States has a unique value of 79 in OpenMarket's network. Please refer to <http://network.OpenMarket.com/> for a complete real-time list of all carriers and their network identification codes.

For example, a user on Cingular in the United States sends an SMS to your short code. The carrier id for Cingular is 383 and OpenMarket will include this code within the XML that it sends to your system. If you parse the XML using one of OpenMarket's SMS SDKs, the carrier id will be available in a property known as the "Source Carrier". Please refer to the sample code below for information on how to retrieve this value.

The carrier id is valuable for numerous reasons. It can also be used to customize your application in order to meet a specific carrier requirement.

### HTTP Delivery

OpenMarket will forward all received SMS messages as XML packets to a URL of your choosing. For example, OpenMarket will forward SMS sent to short code 10435 to:

<http://www.yourcompany.com/OpenMarket/receive.asp>

The XML packet will be sent as an HTTP POST as form-encoded data. This data will be contained within the "xml" parameter of the form post. Your application should extract out this data and use it as a parameter to the Parse() method in the SDK.

---

At the current time, OpenMarket will treat an inbound SMS as successfully forwarded to your system if your web server returns an HTTP response of 200. Our network will not check the content of the HTTP POST, rather it will only check to make sure that the HTTP response code is 200.

The following examples will illustrate how this can be achieved in common web programming languages such as ASP or JSP (please see the examples directory contained with each kit install for more information):

### ActiveX SMS SDK, Microsoft IIS, Active Server Pages

```
Set sms = Server.CreateObject("OpenMarket.SMS")
If (sms.Parse(Request("xml")) And sms.IsDeliver) Then
    <access properties of message>
    <for example>
    ' the sender of the SMS
    Response.Write("source type: " & sms.SourceType)
    Response.Write("source addr: " & sms.SourceAddr)
    Response.Write("source carrier: " & sms.SourceCarrier)
    // the short code sent to
    Response.Write("dest type: " & sms.DestinationType)
    Response.Write("dest addr: " & sms.DestinationAddr)
    Response.Write("msg text: " & sms.MsgText)
    <rest of code>
Else
    <handle invalid xml parsing>
End If
<rest of code>
```

### Java SMS SDK, Apache, Java Server Pages

```
SMS sms = new SMS();
if (sms.parse(request.getParameter("xml")) && sms.isDeliver()) {
    //<access properties of message>
    //<for example>
    // the sender of the SMS
    Address source = sms.getSourceAddr();
    response.out.println("source addr: " + source);
    response.out.println("source carrier: " +
source.getCarrier());
    // the short code sent to
    Address dest = sms.getDestinationAddr();
    response.out.println("dest addr: " + dest);
    response.out.println("msg text: " + sms.getMsgText());
    //<rest of code>
```

```

} else {
    //<handle invalid xml parsing>
}
//<rest of code>

```

## Perl SMS SDK, Apache, CGI Script

```

$sms = Net::SMS->new();
# parse xml and handle delivered message
if ($sms->parse($xml) && $sms->isDeliver()) {
    print "user agent: " . $sms->userAgent . "\n";
    print "ticket id: " . $sms->ticketId . "\n";

    print "src carrier: " . $sms->sourceCarrier . "\n";
    print "src type: " . $sms->sourceType . "\n";
    print "src addr: " . $sms->sourceAddr . "\n";

    print "dest carrier: " . $sms->destinationCarrier . "\n";
    print "dest type: " . $sms->destinationType . "\n";
    print "dest addr: " . $sms->destinationAddr . "\n";

    print "msg text: " . $sms->msgText . "\n";
} else {
    # handle invalid message
}

```

## PHP SMS SDK, Apache, PHP Script

```

require_once("SMS.php");
$sms = new SMS();
// PHP "magic quotes" option will add slashes to FORM params
// which will cause a parsing error -- slashes need removed first
if (get_magic_quotes_gpc()) {
    $xml = stripslashes($_POST['xml']);
} else {
    $xml = $_POST['xml'];
}
// parse incoming xml and check if its a deliver request
if ($sms->parse($xml) && $sms->isDeliver()) {
    echo "User Agent: " . $sms->getUserAgent() . "<br>\n";
    echo "Ticket Id: " . $sms->getTicketId() . "<br>\n";

    $source = $sms->getSourceAddr();
    echo "Source Type: " . $source->getType() . "<br>\n";
    echo "Source Address: " . $source->getAddress() . "<br>\n";

    $dest = $sms->getDestinationAddr();
    echo "Dest Type: " . $dest->getType() . "<br>\n";
    echo "Dest Address: " . $dest->getAddress() . "<br>\n";
    echo "Message Text: " . $sms->getMessageText() . "<br>\n";
} else {

```

---

```
} // handle invalid message
```

---

## Support

Please submit any problems, bug reports, incompatibilities, requests for change, or other comments to [OpenMarket Support](#). All bug reports should be accompanied by one or more concrete examples that will help OpenMarket reproduce the problem. Include all relevant information that will help the support staff recreate the particular environment in which the bug was observed.