



Cellact Large Account Web Service v.3.5

**Interface Description Document
Revision August 2012**

Important Notice

This document is delivered subject to the following conditions and restrictions:

- ☐ This document contains proprietary information belonging to Cellact Ltd. Such information is supplied solely for the purpose of assisting explicitly and properly authorized users of Cellact products.
- ☐ No part of this document may be used for any other purpose, disclosed to any person or firm or reproduced by any means, electronic or mechanical, without the express prior written permission of Cellact Ltd.
- ☐ The software described in this document is furnished under a license. The Cellact software may be used or copied only in accordance with the terms of that agreement.
- ☐ Information contained in this document, specifications on which the document is based and the examples contained in it are all subject to change without notice.
- ☐ Corporate and individual names and data used in examples herein are fictitious unless otherwise noted. The text and the graphics included in this document are for the purposes of instruction, illustration and reference only.
- ☐ The names of other companies, product brands, and services mentioned in this document are trademarks or registered trademarks of their respective holders.

Copyright ©2000, Cellact Ltd. All rights reserved.

Contents

1. Preface	6
1.1 Purpose and Scope	6
1.2 Audience	6
2. Large Account Overview	7
2.1 System Architecture.....	7
2.2 Requirements.....	7
3. Protocol Description	9
3.1 SOAP Structure and Parameters.....	9
3.2 Required Account Details and Verification	9
3.3 Authentication Mechanism.....	10
3.4 Billing Models (SMS)	10
3.5 Message Length Limitations	11
4. Send Procedure.....	13
4.1 Testing	13
4.2 Code Example	13
4.3 Result Message	17
4.4 Using the Script.....	18
4.4.1 Procedure	18
4.4.2 Code Example	18
4.5 Commands.....	21
4.5.1 Sendtextmt.....	21
4.5.2 Sendlinkmt	22
4.5.3 Sendmms.....	24
5. Send Text Procedure	26
5.1 Testing	26
5.2 Code Example	26
5.3 Result Message	28
5.4 Using the Script.....	29
5.4.1 Procedure	29
5.4.2 Code Example	29
6. Send e-mail	33
6.1 Templates	33
6.2 Group management (optional).....	33
6.3 E-mail Web service	33

7. Confirmation on Delivery.....	36
7.1 Confirmation Request	36
7.1.1 MT Event States	38
7.1.2 Confirmation: Message was Delivered to GW	39
7.1.3 Confirmation: Message did not Reach GW	39
7.1.4 Confirmation: Message was Delivered	40
7.1.5 Confirmation: Message was not Delivered	40
8. Reverse Billing Model	41
9. Appendices	44
9.1 Code example in .net.....	44
9.2 Acronyms	46

List of Tables

Table 1: Message Length Limitations per Operator	11
Table 2: SEND Request Message Parameters.....	15
Table 3: SEND Response Message Fields.....	17
Table 4: SEND TEXT Request Message Parameters	27
Table 5: SEND TEXT Result Message Fields.....	28
Table 6: SEND Request Message Parameters.....	35
Table 7: EVT Tags	38
Table 8: Acronyms	46

1. Preface

1.1 Purpose and Scope

This document describes the Cellact Large Account (LA) Web Service interface and SOAP XML configuration. The Large Account application platform provides enterprises and content providers with the ability to send and receive SMS and MMS messages to and from mobile operators, in Israel or abroad. The LA also provides the ability to send marketing e-mail messages based on predefined templates that are managed using the meV4 product.

The Web service URL is: <http://la1.cellactpro.com/SendSms.asmx>

This guide includes the following main topics:

- [Large Account Overview](#)
- [Protocol Description](#)
- [Send Procedure](#)
- [Send Text Procedure](#)
- [Confirmation on Delivery](#)
- [Reverse Billing Model](#)

The appendices include:

- [Acronyms](#)

1.2 Audience

This guide is intended for the mobile provider's system operators. It is assumed that users of the interface are familiar with Cellact Large Account systems and with Web Service usage.

2. Large Account Overview

The Large Account (LA) application platform provides enterprises and content providers with the ability to send and receive SMS and MMS messages to and from mobile operators, in Israel or abroad.

The LA provides several types of interfaces including HTTP/XML, Web Service and SMTP.

The scope of this protocol explains the use of the Web Service interface.

2.1 System Architecture

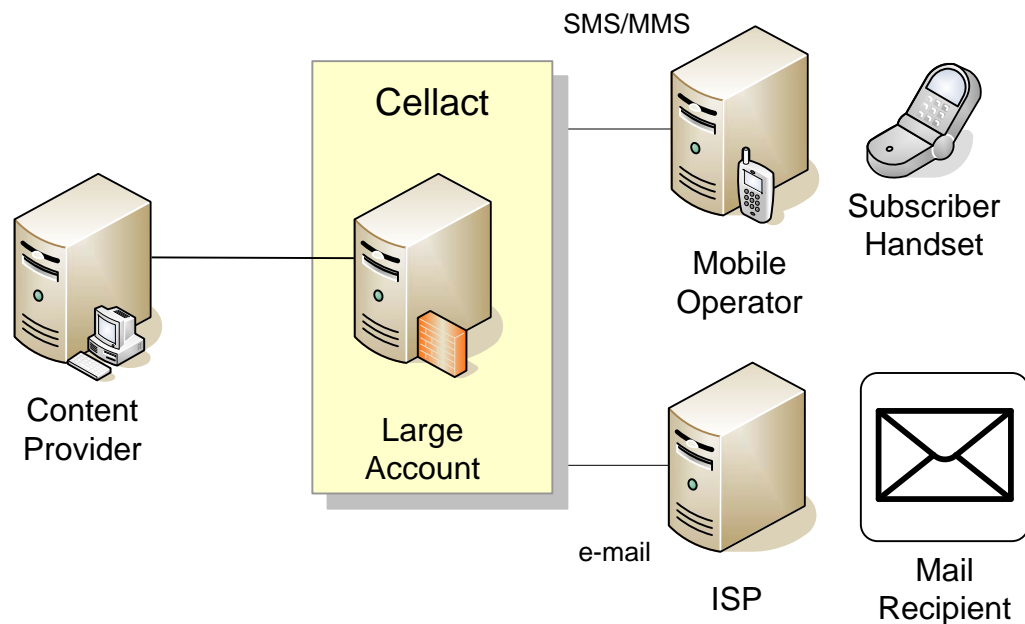


Figure 1: Large Account System Architecture

2.2 Requirements

The following are the basic requirements for sending messages using the Large Account application.

You should ensure that the following actions are performed on the client side:

- Open a Large Account on the Web server Large Account (LA) application.
- Provide a well-known legal IP address on the Internet from where the HTTP/POST requests can be sent to the Large Account application.



When the Large Account server receives a message from the client, it checks if there is a match between the message's IP address and the address of the company that sent the message.

- Assign a unique enterprise name identifier as registered with the LA application.

- Assign a user name and password to the LA application.
- Connect to the Internet.
- Check if the connection to the Internet is established via a TCP/IP port 80 or via the HTTP protocol (firewall rules).
- SOAP Toolkit version 3 or higher must be installed.

3. Protocol Description

3.1 SOAP Structure and Parameters

The LA Web Service protocol requires a HTTP post request with a specific XML format in order to send a message. The message uses the POST method with a single parameter named: "XMLSchema".

The XMLSchema parameter contains the XML string that is used in the request.

The SOAP XML format includes an Envelope section and a Body section, which includes authentication fields, recipients, command, content and optional parameters.

3.2 Required Account Details and Verification

Once you open a new account, you should ensure that you have the following information:

- **Account:** A new account is opened for you on the Cellact Large Account gateway. Cellact provides the username, password and company data necessary for account authentication.
- **IP address:** A valid and well-known IP address of the account, from which the HTTP/POST requests are sent via Internet to the Large Account gateway.



**This address must be provided to Cellact's support personnel.
To contact Cellact, call us or write to support@cellact.com.**

When the Large Account server receives a message from a client, it verifies if there is a match between the message's IP address and that of the company that sent the message. The parameters used for matching are:

- **Unique enterprise name identifier** as registered with the LA application. This is the company element.
- **Username and password** of the Large Account.
- Cellact also requires **connectivity** to the Internet via port 80 with HTTP protocol or port 443 using HTTPS.



While the system accepts requests that are in other formats, they are rejected before they reach the SMSC of the mobile operator, and are not delivered.



The sender's phone number (Call back number) may be any valid telephone number or short number (for example +972506123456 or 1234). This number appears in the Reply field on the target device.

3.3 Authentication Mechanism

The Authentication mechanism identifies the incoming requests. Authentication is based on the following parameters:

- Username attribute
- Password attribute
- Company attribute
- IP address



The incoming requests should always have a fixed IP address.

3.4 Billing Models (SMS)

The following types of billing models are available:

- **Regular message:** The content provider is charged for every message it sends. This is the regular message model.
- **Reverse Billing message:** Reverse Billing. Premium charge – the subscriber that receives the message is charged.
For more information, see [Reverse Billing Model](#).



- In the Orange network, there is no option to use Reverse Billing messages for prepaid subscribers.
- In the Mircs network, it is not possible to set the Sender field. In this case it is replaced with a fixed number that is used for billing purposes.

3.5 Message Length Limitations

Operator support for SMS varies, depending on the maximum allowed message length and on message concatenation support. These two factors influence the SMS setup and display, as well as billing.

Operators use two methods to process messages that exceed the preset text limit:

- **Concatenation:** Concatenation is a system feature that allows for dividing and sending a longer SMS so it is displayed as two or three pages on the handset screen. Concatenated messages are charged according to the length of the message.
- **Splitting:** Operators who do not support concatenation use the split method where messages longer than the set limit arrive at the subscriber's handset as separate messages, not as a single 2-3-page concatenated message. The split messages are charged independently.



A Message that contains one or more non-Latin characters are billed and considered as a Unicode message.

Table 1 lists the concatenation support and the supported maximum message length per operator in characters for each type of interface (Unicode or English).

Table 1: Message Length Limitations per Operator

Operator	Concatenation Support	Unicode (non-Latin, incl. Hebrew, mixed)		English (Latin chars only)	
		Single	Concat.	Single	Concat.
Cellcom	Yes	70	800	160	800
Mirs	No – (split)	70	132	160	-
Orange	Yes	70	800	160	800
Pelephone	No – CDMA (split)	70	800	160	800
Pelephone	Yes – UMTS	70	800	160	800
Home Cellolar	Yes	70	800	160	800
Rami Levi	Yes	70	800	160	800
Golan Telecom	Yes	70	800	160	800
Hot Mobile	Yes	70	800	160	800

Outside Israel (GSM networks)	No	70 characters	800	160	800
----------------------------------	----	------------------	-----	-----	-----



In Telephone network there are still CDMA subscribers that work with 126 characters messages for both English and Hebrew. Charging in such case is still based on the GSM model. For example a 126 message in Hebrew on CDMA subscriber is counted as 2 messages.

4. Send Procedure

Send mobile terminated messages.

Send messages to a set of specified destination addresses.

The Web service URL is: <http://la1.cellactpro.com/SendSms.asmx>

4.1 Testing

The test form is only available for requests from the local machine.

4.2 Code Example

The following is an example of a “**Send**” request and response using SOAP.



The **placeholders** shown here in blue should be replaced with actual values.

```
POST /SendSms.asmx HTTP/1.1
Host: la1.cellactpro.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.cellact.com/webservices/Send"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Send xmlns="http://www.cellact.com/webservices/">
      <credentials>
        <username>string</username>
        <password>string</password>
        <company>string</company>
      </credentials>
      <sendRequest>
        <transactionId>string</transactionId>
        <application>string</application>
        <command>string</command>
        <timeToSend>int</timeToSend>
        <timeToLive>int</timeToLive>
        <deliveryAddresses>
          <DeliveryReportAddress>
            <type>none or email or http</type>
            <address>string</address>
          </DeliveryReportAddress>
        </deliveryAddresses>
        <sender>string</sender>
        <content>string</content>
```

```
<link>string</link>
<destinationAddresses>
  <DestinationAddress>
    <address>string</address>
  </DestinationAddress>
  <DestinationAddress>
    <address>string</address>
  </DestinationAddress>
</destinationAddresses>
<links>
  <string>string</string>
  <string>string</string>
</links>
<messageId>string</messageId>
<serviceName>string</serviceName>
<serviceCode>string</serviceCode>
</sendRequest>
</Send>
</soap:Body>
</soap:Envelope>
```

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: **length**

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SendResponse xmlns="http://www.cellact.com/webservices/">
      <SendResult>
        <result>boolean</result>
        <sessionId>string</sessionId>
        <errorDescription>string</errorDescription>
      </SendResult>
    </SendResponse>
  </soap:Body>
</soap:Envelope>
```

Table 2 describes the parameters included in the SEND message XML.
All fields that are not defined as optional are mandatory.

Table 2: SEND Request Message Parameters

Field	Description
Credentials	The set of identification parameters of the message.
username & password	User and password of the application provided by Cellact.
company	Enterprise unique name provided by Cellact.
sendRequest	Command-related data in free text format.
transactionId	This field is optional. A unique 128-bit number generated by the Large Account server that identifies and closes a transaction. The client should send this ID with the request for a mobile terminated (MT) message.
application	The name of the application that produced the message. In this case the application name is: LA .
command	Any command or command alias.
timeToSend	This field is optional. Time to Send (TTS) is used to schedule a message. This determines how much time can pass before the message is sent to the operator's SMSC. This is a configurable parameter per account. The default limit is 10080 minutes (7 days) from the current date.
TimeToLive	This field is optional. Time to Live (TTL) is used to set the message expiry date. The period of time that the operator's SMSC retains a message in case of delivery failure before the message is expired. The default limit is 4320 minutes (3 days). The minimum limit is 15 minutes.
deliveryAddresses	This field is optional. A set of URL addresses that receive delivery notifications.
DeliveryReportAddress	This field is optional. The URL or Email that receives notifications from the Large Account on message delivery to the subscriber's handset.

Field	Description
Type	<p>This field is optional. Specifies the delivery notification format:</p> <ul style="list-style-type: none">■ None: No notification is sent.■ Email: Notifications is sent via email.■ HTTP: Notifications is sent via http.
address	<p>This field is optional. The URL or Email for receiving notifications.</p>
sender	<p>Callback number of the sender.</p>
content	<p>The message text.</p> <p>The content limit (maximum message length) is 470 characters. The entire message is blocked if the content exceeds this limit.</p> <p>Note: Message length limits vary and must be specified according to the type of the destination mobile network. For more information, see Message Length Limitations.</p>
link	<p>This field is optional. Used to specify the link in "sendlinkmt", "sendringtonemt" and "sendiconmt" (Mobile Terminated) messages.</p>
DestinationAddress	<p>A list of all destination numbers for the message in the international (not national) format.</p> <p>Note: A request may contain up to 60 destination numbers.</p>
address	<p>A single destination number of a message.</p>
links	<p>This field is optional. Set of links for MMS messages.</p>
string	<p>This field is optional. A link for an MMS message.</p>
messageld	<p>This field is optional. A unique identification provided by the client to sign the message. This field is included in the notification.</p>
serviceName	<p>This field is optional. A field for client's use. May contain text. The result message includes this information.</p>
serviceCode	<p>This field is optional. The preset Service Code for charging the transaction.</p>

4.3 Result Message

In response to the Web Service request made by the enterprise client (CP), the Large Account server returns a result.

If the request is successful, the following format is returned:

```
<SendTextResult>
  <result>true</result>
  <sessionId>c05b21a4-5d7f-4c99-af2e-141b61b2856c</sessionId>
</SendTextResult>
```

In case the request fails, the following format is returned:

```
<SendTextResult>
  <result>false</result>
  <ErrorCode>not an authorized user</ErrorCode>
</SendTextResult>
```

Table 3 describes the fields included in the returned result XML. All parameters that are not defined as optional are mandatory.

Table 3: SEND Response Message Fields

Field	Description
SendResponse	The root tag of the message.
sessionId	A unique 128-bit number generated by the Large Account server that provides a way to identify and further be notified on the status of a request.
result	If the request is in the correct format, the result is: true . If the request fails, the result is: false .
errorCode	This field is optional. Returns the code of the error that occurred.

4.4 Using the Script

4.4.1 Procedure

- 1 Copy the following code to a **VBS** file, and test the application.
- 2 Replace the **username**, **password** and **company Credentials** in the code.
 - **XXXUSERNAMEXXX**
 - **XXXPASSWORDXXX**
 - **XXXCOMPANYXXX**
- 3 Specify the **Destination Number**, mentioned as +9725XXXXXXXXX in the code.



A Destination Number in Israel must begin with +972 opposed to the regular API which does not require this area code.

- 4 Run the script.

4.4.2 Code Example

The following is the full script to run.

```
Option Explicit

Private Const Method = "Send"
Private Const END_POINT_URL =
"http://lal.cellactpro.com/SendSms.asmx"
Private Const CALC_NS = "http://www.cellact.com/webservices/"

Const Count = 1
On Error Resume Next
Main
If Err.Number <> 0 Then
    WScript.Echo Err.Description
End If

WScript.Quit

Private Sub Main()
Dim Serializer          'As SoapSerializer30
Dim Reader              'As SoapReader30
Dim Connector           'As SoapConnector30
Set Serializer = WScript.CreateObject("MSSOAP.SoapSerializer30")
Set Connector = WScript.CreateObject("MSSOAP.HttpConnector30")
Set Reader = WScript.CreateObject("MSSOAP.SoapReader30")
```

```

Dim I
For I=1 To Count
Connector.Property("EndPointURL") = END_POINT_URL
Connector.Connect
Connector.Property("SoapAction") = SoapAction '
binding/operation/soapoperation
Connector.BeginMessage
Serializer.Init Connector.InputStream
CreateRequest Serializer
Connector.EndMessage

Reader.Load Connector.OutputStream
ProcessResponse Reader
Next
End Sub

Private Sub Echo (Message)
WScript.Echo Message
End Sub

Private Function SoapAction()
soapAction = CALC_NS & Method
End Function

Private Sub ProcessResponse (Reader)
If Not Reader.Fault Is Nothing Then
WScript.Echo "Fault:" & Reader.Fault.Xml
End If

'Echo Reader.Envelope.Xml
'Echo Reader.Body.Xml
'Echo Reader.RPCResult.Xml

Dim SendTextResultElement
Set SendTextResultElement = Reader.RPCParameter (Method &
"Result", CALC_NS )
If SendTextResultElement Is Nothing Then Exit Sub
SendTextResultElement.ownerDocument.SetProperty
"SelectionNamespaces",
"xmlns:my=""http://www.cellact.com/webservices/""
Dim ResultString
ResultString = ResultString & "result: "
If Not SendTextResultElement.SelectSingleNode ( "my:result") Is
Nothing Then
ResultString = ResultString &
SendTextResultElement.SelectSingleNode ( "my:result").Text
End If

ResultString = ResultString & VbCrLf
ResultString = ResultString & "errorDescription: "

```

```
If Not SendTextResultElement.SelectSingleNode (
"my:errorDescription") Is Nothing Then
ResultString = ResultString &
SendTextResultElement.SelectSingleNode (
"my:errorDescription").Text
End If

ResultString = ResultString & VbCrLf
ResultString = ResultString & "sessionId: "
If Not SendTextResultElement.SelectSingleNode ( "my:sessionId")
Is Nothing Then
ResultString = ResultString &
SendTextResultElement.SelectSingleNode ( "my:sessionId").Text
End If

ResultString = ResultString & VbCrLf
Echo ResultString
End Sub

Sub CreateRequest ( Serializer)
Serializer.StartEnvelope
Serializer.StartBody
Serializer.StartElement Method, CALC_NS
Serializer.SoapDefaultNamespace CALC_NS
Serializer.StartElement "credentials"
Serializer.StartElement "username"
Serializer.WriteString "XXXUSERNAMEXXX"
Serializer.EndElement

Serializer.StartElement "password"
Serializer.WriteString "XXXPASSWORDXXX"
Serializer.EndElement

Serializer.StartElement "company"
Serializer.WriteString "XXXCOMPANYXXX"
Serializer.EndElement 'company
Serializer.EndElement

Serializer.StartElement "sendRequest"
Serializer.StartElement "destinationAddresses"
Serializer.StartElement "DestinationAddress"
Serializer.StartElement "address"
Serializer.WriteString "+9725XXXXXXXXX"
Serializer.EndElement 'address
Serializer.EndElement 'DestinationAddress
Serializer.EndElement 'destinationAddresses

Serializer.StartElement "command"
Serializer.WriteString "sendtextmt"
Serializer.EndElement 'command
```

```

Serializer.StartElement "content"
Serializer.WriteString "testing web service"
Serializer.EndElement 'content
Serializer.EndElement
Serializer.EndElement
Serializer.EndBody
Serializer.EndEnvelope
End Sub

```

4.5 Commands

The following message commands are available in the Send Procedure of the LA Web Service v.3.3:

Sendtextmt	Send a mobile terminated SMS message.
Sendlinkmt	Send a mobile terminated WAP link message.
Sendmms	Send a mobile terminated MMS message.

4.5.1 Sendtextmt



The request's destinationAddresses list may contain up to 60 destination numbers.

Entering the **sendtextmt** command in the command element of the request sends a text message to the list of subscribers on the destinationAddresses list.

Following is an example of the Envelope and Body sections of the request In VB script:

```

Sub CreateRequest ( Serializer)
Serializer.StartEnvelope
Serializer.StartBody
Serializer.StartElement Method, CALC_NS
Serializer.SoapDefaultNamespace CALC_NS
Serializer.StartElement "credentials"
Serializer.StartElement "username"
Serializer.WriteString "XXXUSERNAMEXXX"
Serializer.EndElement

Serializer.StartElement "password"
Serializer.WriteString "XXXPASSWORDXXX"
Serializer.EndElement

Serializer.StartElement "company"

```

```
Serializer.WriteString "XXXCOMPANYXXX"
Serializer.EndElement 'company
Serializer.EndElement

Serializer.StartElement "sendRequest"
Serializer.StartElement "destinationAddresses"
Serializer.StartElement "DestinationAddress"
Serializer.StartElement "address"
Serializer.WriteString "+9725XXXXXXXXX"
Serializer.EndElement 'address
Serializer.EndElement 'DestinationAddress
Serializer.EndElement 'destinationAddresses

Serializer.StartElement "command"
Serializer.WriteString "sendtextmt"
Serializer.EndElement 'command

Serializer.StartElement "sender"
Serializer.WriteString "1234"
Serializer.EndElement 'sender

Serializer.StartElement "content"
Serializer.WriteString "testing web service text"
Serializer.EndElement 'content
Serializer.EndElement
Serializer.EndElement
Serializer.EndBody
Serializer.EndEnvelope
End Sub
```

4.5.2 Sendlinkmt



The following limitations apply to sendlinkmt requests:

- The length of the URL string (assuming use of Latin characters only, such as English) may contain up to 200 characters.
- The content may contain up to 201 characters.
- The request's destinationAddresses list may contain up to 60 destination numbers.

Entering the **sendlinkmt** command in the command element of the request sends a URL link message (usually used for WAP links) to the list of subscribers on the destinationAddresses list.

Following is an example of the Envelope and Body sections of the request In VB script:

```
Sub CreateRequest ( Serializer)
Serializer.StartEnvelope
Serializer.StartBody
Serializer.StartElement Method, CALC_NS
Serializer.SoapDefaultNamespace CALC_NS
Serializer.StartElement "credentials"
Serializer.StartElement "username"
Serializer.WriteString "XXXUSERNAMEXXX"
Serializer.EndElement

Serializer.StartElement "password"
Serializer.WriteString "XXXPASSWORDXXX"
Serializer.EndElement

Serializer.StartElement "company"
Serializer.WriteString "XXXCOMPANYXXX"
Serializer.EndElement 'company
Serializer.EndElement

Serializer.StartElement "sendRequest"
Serializer.StartElement "destinationAddresses"
Serializer.StartElement "DestinationAddress"
Serializer.StartElement "address"
Serializer.WriteString "+9725XXXXXXXXX"
Serializer.EndElement 'address
Serializer.EndElement 'DestinationAddress
Serializer.EndElement 'destinationAddresses

Serializer.StartElement "command"
Serializer.WriteString "sendlinkmt"
Serializer.EndElement 'command

Serializer.StartElement "sender"
Serializer.WriteString "1234"
Serializer.EndElement 'sender

Serializer.StartElement "content"
Serializer.WriteString "testing web service link"
Serializer.EndElement 'content
Serializer.StartElement "link"
Serializer.WriteString "http://www.google.com"
Serializer.EndElement 'link
Serializer.EndElement
Serializer.EndElement
Serializer.EndBody
Serializer.EndEnvelope
End Sub
```

4.5.3 Sendmms



- Each mobile operator may support a different combination of MMS file formats, see also [MMS Support](#).
- The request's destinationAddresses list may contain up to 60 destination numbers.

Entering the **sendmms** command in the command element of the request sends a MMS message to the subscribers on the destinationAddresses list.

Sending an MMS message includes a list of files that are sent to the mobile device, currently using a simple template of the content.

Following is an example of the Envelope and Body sections of the request In VB script:

```
Sub CreateRequest ( Serializer)
  Serializer.StartEnvelope
  Serializer.StartBody
  Serializer.StartElement Method, CALC_NS
  Serializer.SoapDefaultNamespace CALC_NS
  Serializer.StartElement "credentials"
  Serializer.StartElement "username"
  Serializer.WriteString "XXXUSERNAMEXXX"
  Serializer.EndElement

  Serializer.StartElement "password"
  Serializer.WriteString "XXXPASSWORDXXX"
  Serializer.EndElement

  Serializer.StartElement "company"
  Serializer.WriteString "XXXCOMPANYXXX"
  Serializer.EndElement 'company
  Serializer.EndElement

  Serializer.StartElement "sendRequest"
  Serializer.StartElement "destinationAddresses"
  Serializer.StartElement "DestinationAddress"
  Serializer.StartElement "address"
  Serializer.WriteString "+9725XXXXXXXXXX"
  Serializer.EndElement 'address
  Serializer.EndElement 'DestinationAddress
  Serializer.EndElement 'destinationAddresses

  Serializer.StartElement "command"
  Serializer.WriteString "sendmms"
  Serializer.EndElement 'command
```



```
Serializer.StartElement "sender"
Serializer.WriteString "1234"
Serializer.EndElement 'sender

Serializer.StartElement "content"
Serializer.WriteString "testing web service MMS"
Serializer.EndElement 'content
Serializer.StartElement "links"
Serializer.StartElement "string"
Serializer.WriteString "http://www.cellact.com/icon.gif"
Serializer.EndElement 'string
Serializer.StartElement "string"
Serializer.WriteString "http://www.cellact.com/ringtone.mid"
Serializer.EndElement 'string
Serializer.StartElement "string"
Serializer.WriteString "http://www.cellact.com/text.txt"
Serializer.EndElement 'string
Serializer.EndElement 'links
Serializer.EndElement
Serializer.EndElement
Serializer.EndBody
Serializer.EndEnvelope
End Sub
```

4.5.3.1 MMS Support

MMS support by the mobile operators is as follows:

- **Peleephone** supports some format combinations.
- **Orange** requires an advance notification of combination MMS messages, and supports only some format combinations.
- **Cellcom** allows combinations of different MMS file formats without any limitations.
- **Mirs** does not support MMS.

5. Send Text Procedure

Send a mobile terminated SMS message.

Send a simple text message to a set of specified destination addresses.

The Web service URL is: <http://la1.cellactpro.com/SendSms.aspx>

5.1 Testing

The test form is only available for requests from the local machine.

5.2 Code Example

The following is an example of a **Send Text** request and response using SOAP.



The **placeholders** shown here in blue should be replaced with actual values.

```
POST /SendSms.aspx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.cellact.com/webservices/SendText"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SendText xmlns="http://www.cellact.com/webservices/">
      <credentials>
        <username>string</username>
        <password>string</password>
        <company>string</company>
      </credentials>
      <content>string</content>
      <destinationAddressSet>
        <string>string</string>
        <string>string</string>
      </destinationAddressSet>
    </SendText>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SendTextResponse xmlns="http://www.cellact.com/webservices/">
      <SendTextResult>
        <result>boolean</result>
        <sessionId>string</sessionId>
        <errorCode>string</errorCode>
      </SendTextResult>
    </SendTextResponse>
  </soap:Body>
</soap:Envelope>

```

Table 4 describes the parameters included in the SEND TEXT message. All parameters that are not defined as optional are mandatory.

Table 4: SEND TEXT Request Message Parameters

Field	Description
Credentials	The set of identification parameters of the message.
username & password	User and password of the application provided by Cellact.
company	Enterprise unique name provided by Cellact.
content	The message text. The content limit (maximum message length) is 160 characters. The entire message is blocked if the content exceeds this limit. Note: Message length limits vary and must be specified according to the type of the destination mobile network.
destinationAddressSet	A list of all destination numbers for the message in the international (not national) format. Note: A request may contain up to 60 destination numbers.
String	A single destination number of a message.

5.3 Result Message

In response to the Web Service request made by the enterprise client, the Large Account server returns a result.

If the request is successful, the following format is returned:

```
<SendTextResult>
```

```
    <result>true</result>
    <sessionId>c05b21a4-5d7f-4c99-af2e-141b61b2856c
</sessionId>
</SendTextResult>
```

If the request fails, the following format is returned:

```
<SendTextResult>
    <result>false</result>
    <errorDescription>not an authorized user
</errorDescription>
</SendTextResult>
```

Table 5 describes the fields included in the returned result XML. All parameters that are not defined as optional are mandatory.

Table 5: SEND TEXT Result Message Fields

Field	Description
SendTextResult	The root tag of the message.
result	If the request is in the correct format, the result is: true . If the request fails, the result is: false .
errorDescription	This field is optional. Describes the error that occurred.
sessionId	A unique 128-bit number generated by the Large Account server that provides a way to identify and further be notified on the status of a request.

5.4 Using the Script

5.4.1 Procedure

To run a script:

- 1 Copy the following code to a **VBS** file, and test the application.
- 2 Replace the **username**, **password** and **company credentials** in the code.
 - **XXXUSERNAMEXXX**
 - **XXXPASSWORDXXX**
 - **XXXCOMPANYXXX**
- 3 Specify the **Destination Number**, mentioned as +9725XXXXXXXXX in the code.
- 4 Run the script.

5.4.2 Code Example

The following is the full script to run.

```
Option Explicit
```

```
Private Const Method = "Send"
Private Const END_POINT_URL =
"http://lal.cellactpro.com/SendSms.asmx"
Private Const CALC_NS = "http://www.cellact.com/webservices/"
```

```
Const Count = 1
On Error Resume Next
Main
If Err.Number <> 0 Then
    WScript.Echo Err.Description
End If
```

```
WScript.Quit
```

```
Private Sub Main()
Dim Serializer          'As SoapSerializer30
Dim Reader              'As SoapReader30
Dim Connector           'As SoapConnector30
Set Serializer = WScript.CreateObject("MSSOAP.SoapSerializer30")
Set Connector = WScript.CreateObject("MSSOAP.HttpConnector30")
Set Reader = WScript.CreateObject("MSSOAP.SoapReader30")
Dim I
For I=1 To Count
Connector.Property("EndPointURL") = END_POINT_URL
Connector.Connect
```

```
Connector.Property("SoapAction") = SoapAction '
binding/operation/soapoperation
Connector.BeginMessage
Serializer.Init Connector.InputStream
CreateRequest Serializer
Connector.EndMessage

Reader.Load Connector.OutputStream
ProcessResponse Reader
Next
End Sub

Private Sub Echo (Message)
WScript.Echo Message
End Sub
Private Function SoapAction()
soapAction = CALC_NS & Method
End Function

Private Sub ProcessResponse (Reader)
If Not Reader.Fault Is Nothing Then
WScript.Echo "Fault:" & Reader.Fault.Xml
End If

'Echo Reader.Envelope.Xml
'Echo Reader.Body.Xml
'Echo Reader.RPCResult.Xml

Dim SendTextResultElement
Set SendTextResultElement = Reader.RPCParameter (Method &
"Result", CALC_NS )
If SendTextResultElement Is Nothing Then Exit Sub
SendTextResultElement.ownerDocument.SetProperty
"SelectionNamespaces",
"xmlns:my=" & "http://www.cellact.com/webservices/" & ""
Dim ResultString
ResultString = ResultString & "result: "
If Not SendTextResultElement.SelectSingleNode ( "my:result") Is
Nothing Then
ResultString = ResultString &
SendTextResultElement.SelectSingleNode ( "my:result").Text
End If

ResultString = ResultString & VbCrLf
ResultString = ResultString & "errorDescription: "
If Not SendTextResultElement.SelectSingleNode (
"my:errorDescription") Is Nothing Then
ResultString = ResultString &
SendTextResultElement.SelectSingleNode (
"my:errorDescription").Text
End If
```

```
ResultString = ResultString & VbCrLf
ResultString = ResultString & "sessionId: "
If Not SendTextResultElement.SelectSingleNode ( "my:sessionId")
Is Nothing Then
ResultString = ResultString &
SendTextResultElement.SelectSingleNode ( "my:sessionId").Text
End If

ResultString = ResultString & VbCrLf
Echo ResultString
End Sub

Sub CreateRequest ( Serializer)
Serializer.StartEnvelope
Serializer.StartBody
Serializer.StartElement Method, CALC_NS
Serializer.SoapDefaultNamespace CALC_NS
Serializer.StartElement "credentials"
Serializer.StartElement "username"
Serializer.WriteString "XXXUSERNAMEXXX"
Serializer.EndElement

Serializer.StartElement "password"
Serializer.WriteString "XXXPASSWORDXXX"
Serializer.EndElement

Serializer.StartElement "company"
Serializer.WriteString "XXXCOMPANYXXX"
Serializer.EndElement 'company
Serializer.EndElement

Serializer.StartElement "sendRequest"
Serializer.StartElement "destinationAddresses"
Serializer.StartElement "DestinationAddress"
Serializer.StartElement "address"
Serializer.WriteString "+9725XXXXXXXXX"
Serializer.EndElement 'address
Serializer.EndElement 'DestinationAddress
Serializer.EndElement 'destinationAddresses

Serializer.StartElement "content"
Serializer.WriteString "testing web service"
Serializer.EndElement 'content
Serializer.EndElement
Serializer.EndElement
Serializer.EndBody
Serializer.EndEnvelope
End Sub
```


6. Send e-mail

Sending e-mail to a recipient is based on the meV4 capabilities. It means that a user can load predefined e-mail templates and list of recipients – and then send e-mail messages using those templates.

6.1 Templates

Template management takes place on the meV4 application. A user can load different e-mail templates that are going to be used by the API.
















Templates list <input checked="" type="radio"/> Email <input type="radio"/> MMS						
Name		From		To		
<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="button" value="Search"/>
					<input type="button" value="Header/Footer"/>	<input type="button" value="New template"/>
Creation date	Name	Description			Update date	
08/07/2011 11:57:40	David test				08/07/2011 15:02:59	  
08/04/2011 10:16:29	ian				08/04/2011 10:16:29	  
06/28/2011 10:46:23	test remove				08/03/2011 11:47:51	  
05/23/2011 12:50:56	test2				05/23/2011 12:50:56	  
05/19/2011 12:06:05	test				05/19/2011 12:06:05	  
1 2						

Figure 2: Large Account System Architecture

6.2 Group management (optional)

A user can also manage his list of recipients and groups either by the GUI or using web service based interface. The interface is located on <http://ws.cellactpro.net/>

The interface is opened only for specific IP address upon request to support.

This interface includes the ability to:

1. Manage recipients.
2. Manage groups.
3. Block e-mail recipient (Guardian).

There are also other options on this web services for managing the system using API. You can use the GUI to get the same results as the API.

6.3 E-mail Web service

In order to send an e-mail message the API is used for the following options:

1. Send an e-mail to a single recipient that is already defined in the meV4 lists of recipients. This allows the option
2. Send an e-mail to a group of recipients.

The link to sending e-mails is: <http://ws.cellactpro.net/wssend.asmx>

```
POST /wssend.asmx HTTP/1.1
Host: ws.cellactpro.net
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://cellact.com/SendMail"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <WSCredential xmlns="http://cellact.com/">
      <Username>string</Username>
      <Password>string</Password>
      <Company>string</Company>
    </WSCredential>
  </soap:Header>
  <soap:Body>
    <SendMail xmlns="http://cellact.com/">
      <Mail>
        <SenderMail>string</SenderMail>
        <SenderName>string</SenderName>
        <RecipientGroups>
          <string>string</string>
          <string>string</string>
        </RecipientGroups>
        <RecipientMails>
          <string>string</string>
          <string>string</string>
        </RecipientMails>
        <Subject>string</Subject>
        <TemplateName>string</TemplateName>
      </Mail>
    </SendMail>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SendMailResponse xmlns="http://cellact.com/" />
  </soap:Body>
</soap:Envelope>
```

Table 6: SEND Request Message Parameters

Field	Description
Credentials	The set of identification parameters of the message.
username & password	User and password of the application provided by Cellact.
company	Enterprise unique name provided by Cellact.
SendMail	The address of the e-mail sender.
SenderName	The name of the e-mail sender.
ReceipientsGroups	Option to send to a group of recipients according to the alias of meV4.
RecipientsMails	Option to send to recipients e-mail addresses.
Subject	The e-mail subject.
TemplateName	The name of the e-mail template as was loaded in meV4.

7. Confirmation on Delivery

There is an option to request a confirmation on the status of a message, in order to know if the message has reached the subscriber's handset. The confirmation is provided as a **HTTP/GET** request from the Large Account to the URL or Email address of the content provider.

7.1 Confirmation Request



The HTTP confirmation request is always delivered on port 80 and there is no option to receive the message on a different port.

The required field is the DeliveryReportAddress that includes an address destination to the URL address of the server of the content provider. The DeliveryReportAddress is part of the deliveryAddresses element.

In order to receive a confirmation, use the following format:

```
Sub CreateRequest ( Serializer)
Serializer.StartEnvelope
Serializer.StartBody
Serializer.StartElement Method, CALC_NS
Serializer.SoapDefaultNamespace CALC_NS
Serializer.StartElement "credentials"
Serializer.StartElement "username"
Serializer.WriteString "XXXUSERNAMEXXX"
Serializer.EndElement

Serializer.StartElement "password"
Serializer.WriteString "XXXPASSWORDXXX"
Serializer.EndElement

Serializer.StartElement "company"
Serializer.WriteString "XXXCOMPANYXXX"
Serializer.EndElement 'company
Serializer.EndElement

Serializer.StartElement "sendRequest"
Serializer.StartElement "destinationAddresses"
Serializer.StartElement "DestinationAddress"
Serializer.StartElement "address"
Serializer.WriteString "+9725XXXXXXXXX"
Serializer.EndElement 'address
Serializer.EndElement 'DestinationAddress
Serializer.EndElement 'destinationAddresses

Serializer.StartElement "command"
Serializer.WriteString "sendtextmt"
```

```
Serializer.EndElement 'command

Serializer.StartElement "sender"
Serializer.WriteString "1234"
Serializer.EndElement 'sender

Serializer.StartElement "deliveryAddresses"
Serializer.StartElement "DeliveryReportAddress"
Serializer.StartElement "type"
Serializer.WriteString "http"
Serializer.EndElement 'type
Serializer.StartElement "address"
Serializer.WriteString "http://www.hello.com/cod.asp"
Serializer.EndElement 'address
Serializer.EndElement 'DestinationAddress
Serializer.EndElement 'destinationAddresses

Serializer.StartElement "content"
Serializer.WriteString "testing web service confirmation"
Serializer.EndElement 'content
Serializer.EndElement
Serializer.EndElement
Serializer.EndBody
Serializer.EndEnvelope
End Sub
```

In response to the HTTP request generated by the content provider, a HTTP 200 Ok answer is sent with a unique SESSION identifier. This identifier is used to provide the content provider with further information on the status of a message.

When the message arrives to a new state, a notification message is sent by HTTP/GET to the content provider according to the URL address that was provided in the request. The single parameter in the request is called "confirmation" and contains the XML string.

7.1.1 MT Event States

Table 7 lists the messaging event type indicators included in the **EVT** tag of the delivery confirmation message.

Table 7: EVT Tags

EVT	Description
mt_ok	The message has arrived to the mobile operator gateway.
mt_nok	The message did not arrive to the mobile operator gateway; message was blocked.
mt_del	The message has arrived to the mobile handset.
mt_rej	The message did not arrive to the mobile handset and will never reach it.



The Mirc network and some international mobile operators do not support confirmation on delivery.

7.1.2 Confirmation: Message was Delivered to GW

Following is an example of a confirmation that the message has arrived at the mobile operator's gateway (SMSC or MMSC):

CONFIRMATION=

```
<PALO>
  <BLMJ>37f289e3-dca5-436a-8ddc-28257507d48a</BLMJ>
  <SENDER>1234</SENDER>
  <RECIPIENT MNP="97254">+972501234567</RECIPIENT>
  <FINAL_DATE>20080803122112</FINAL_DATE>
  <EVT>mt_ok</EVT>
  <REASON>5000</REASON>
  <MESSAGE_COUNT>1</MESSAGE_COUNT>
</PALO>
```

7.1.3 Confirmation: Message did not Reach GW

Following is an example of a confirmation that the message did not reach the mobile operator's gateway (SMSC or MMSC) and therefore will not reach the mobile handset:

CONFIRMATION=

```
<PALO>
  <BLMJ>d7e78097-4a7f-4893-8ab9-e47f4306fb87</BLMJ>
  <SENDER>0501234567</SENDER>
  <RECIPIENT MNP="97250">+972501234567</RECIPIENT>
  <FINAL_DATE>20080731095516</FINAL_DATE>
  <EVT>mt_nok</EVT>
  <REASON>2010</REASON>
  <MESSAGE_COUNT>1</MESSAGE_COUNT>
</PALO>
```

7.1.4 Confirmation: Message was Delivered

Following is an example of a confirmation that the message has reached the subscriber's handset:

CONFIRMATION=

```
<PALO>

  <BLMJ>a65b9a60-c8c6-46e8-9083-2113e9e98c79</BLMJ>

  <SENDER>0501235467</SENDER>

  <RECIPIENT MNP="97250">+972507124895</RECIPIENT>

  <FINAL_DATE>20080803123145</FINAL_DATE>

  <EVT>mt_del</EVT>

  <REASON>1000</REASON>

  <MESSAGE_COUNT>1</MESSAGE_COUNT>

</PALO>
```

In addition to the mt_del confirmation, a mt_ok confirmation is received from the operator's SMSC/MMSC.

7.1.5 Confirmation: Message was not Delivered

Following is an example of a confirmation that the message was not delivered to the subscriber's handset and will never reach the handset:

CONFIRMATION=

```
<PALO>

  <BLMJ>d7e78097-4565-4893-8ab9-e47f4306fb87</BLMJ>

  <SENDER>0501234567</SENDER>

  <RECIPIENT MNP="97250">+972501234567</RECIPIENT>

  <FINAL_DATE>20080731095516</FINAL_DATE>

  <EVT>mt_rej</EVT>

  <REASON>2010</REASON>

  <MESSAGE_COUNT>1</MESSAGE_COUNT>

</PALO>
```

In addition to the mt_rej confirmation, a mt_ok confirmation is received from the operator's SMSC/MMSC.

8. Reverse Billing Model

The Reverse Billing (RB) model enables the content provider to send a message to a subscriber and charge the subscriber for receiving the message. This is usually at a premium charge of minimum 0.4 NIS per message.

The content provider should indicate the price of the mobile terminated (MT) message by using the serviceCode section. The serviceCode section should indicate the pricing plan in the serviceCode element. The serviceCode element actually sets the price to be charged from the subscriber. Cellact opens a different pricing serviceCode element for every price that is required. The content provider must ensure that it has the correct pricing plan is approved and working.



When using a new pricing plan (serviceCode element) for the first time, send a test message and then verify with Cellact support <mailto:support@cellact.com> that the message was charged correctly.

Following is an example of the Envelope and Body sections of a text message with Reverse Billing request In VB script:

```
Sub CreateRequest ( Serializer)
  Serializer.StartEnvelope
  Serializer.StartBody
  Serializer.StartElement Method, CALC_NS
  Serializer.SoapDefaultNamespace CALC_NS
  Serializer.StartElement "credentials"
  Serializer.StartElement "username"
  Serializer.WriteString "XXXUSERNAMEXXX"
  Serializer.EndElement

  Serializer.StartElement "password"
  Serializer.WriteString "XXXPASSWORDXXX"
  Serializer.EndElement

  Serializer.StartElement "company"
  Serializer.WriteString "XXXCOMPANYXXX"
  Serializer.EndElement 'company
  Serializer.EndElement

  Serializer.StartElement "sendRequest"
  Serializer.StartElement "destinationAddresses"
  Serializer.StartElement "DestinationAddress"
  Serializer.StartElement "address"
  Serializer.WriteString "+9725XXXXXXX"
  Serializer.EndElement 'address
  Serializer.EndElement 'DestinationAddress
  Serializer.EndElement 'destinationAddresses
```

```
Serializer.StartElement "command"
Serializer.WriteString "sendtextmt"
Serializer.EndElement 'command

Serializer.StartElement "sender"
Serializer.WriteString "1234"
Serializer.EndElement 'sender

Serializer.StartElement "content"
Serializer.WriteString "testing web service RB text"
Serializer.EndElement 'content
Serializer.StartElement "serviceCode"
Serializer.WriteString "EXAMPLE1"
Serializer.EndElement 'serviceCode
Serializer.EndElement
Serializer.EndElement
Serializer.EndBody
Serializer.EndEnvelope
End Sub
```

Pay attention to the following fields:

- StartElement "serviceCode": This element indicates a Reverse Billing message.
- WriteString: Inside the serviceCode section, this string indicates the content provider's pricing plan.

Following is another example of a URL link message that is sent using a Reverse Billing model:

```
Sub CreateRequest ( Serializer)
Serializer.StartEnvelope
Serializer.StartBody
Serializer.StartElement Method, CALC_NS
Serializer.SoapDefaultNamespace CALC_NS
Serializer.StartElement "credentials"
Serializer.StartElement "username"
Serializer.WriteString "XXXUSERNAMEXXX"
Serializer.EndElement

Serializer.StartElement "password"
Serializer.WriteString "XXXPASSWORDXXX"
Serializer.EndElement

Serializer.StartElement "company"
Serializer.WriteString "XXXCOMPANYXXX"
Serializer.EndElement 'company
Serializer.EndElement
```

```
Serializer.StartElement "sendRequest"
Serializer.StartElement "destinationAddresses"
Serializer.StartElement "DestinationAddress"
Serializer.StartElement "address"
Serializer.WriteString "+9725XXXXXXX"
Serializer.EndElement 'address
Serializer.EndElement 'DestinationAddress
Serializer.EndElement 'destinationAddresses

Serializer.StartElement "command"
Serializer.WriteString "sendlinkmt"
Serializer.EndElement 'command

Serializer.StartElement "sender"
Serializer.WriteString "1234"
Serializer.EndElement 'sender

Serializer.StartElement "content"
Serializer.WriteString "testing web service RB link"
Serializer.EndElement 'content
Serializer.StartElement "link"
Serializer.WriteString "http://www.google.com"
Serializer.EndElement 'link
Serializer.StartElement "serviceCode"
Serializer.WriteString "EXAMPLE1"
Serializer.EndElement 'serviceCode
Serializer.EndElement
Serializer.EndElement
Serializer.EndBody
Serializer.EndEnvelope
End Sub
```

9. Appendices

9.1 Code example in .net

```
SendSms senderObj = new SendSms ()
credentials = LoadCredentialsObject();
SendRequest request = LoadSendRequestObject();
result = senderObj.Send( credentials, request);

private Credentials LoadCredentialsObject()
{
    Credentials credentials = new Credentials();
    credentials.username = textBoxUsername.Text;
    credentials.password = textBoxPassword.Text;
    credentials.company = textBoxCompany.Text;
    return credentials;
}

private SendRequest LoadSendRequestObject()
{
    SendRequest request = new SendRequest ();
    if (textBoxCommand.TextLength > 0)
    {
        request.command = textBoxCommand.Text;
    }
    if ( textBoxSender.TextLength > 0)
    {
        request.sender = textBoxSender.Text;
    }
    if ( textBoxTransactionId.TextLength > 0)
    {
        request.transactionId = textBoxTransactionId.Text;
    }
    if ( textBoxMessageId.TextLength > 0)
    {
        request.messageId = textBoxMessageId.Text;
    }
    if ( textBoxTimeToSend.TextLength > 0)
    {
        request.timeToSend = Int32.Parse ( textBoxTimeToSend.Text);
    }
    if ( textBoxTimeToLive.TextLength > 0)
    {
        request.timeToLive = Int32.Parse ( textBoxTimeToLive.Text);
    }
    if ( textBoxServiceCode.TextLength > 0)
    {
        request.serviceCode = textBoxServiceCode.Text;
    }
    if ( textBoxServiceName.TextLength > 0)
    {
        request.serviceName = textBoxServiceName.Text;
    }
    if ( textBoxLink.TextLength > 0)
```

```
    {
        request.link = textBoxLink.Text;
    }
    if (textBoxContent.TextLength > 0){
        request.content = textBoxContent.Text;
    }
    request.destinationAddresses = new
DestinationAddress[listBoxPhone.Items.Count];
    for (int i = 0 ; i < this.listBoxPhone.Items.Count ; i++)
    {

        request.destinationAddresses[i] = new DestinationAddress();
        request.destinationAddresses[i].address =
listBoxPhone.Items[i].ToString();
    }
    return request;
}
```

9.2 Acronyms

Table 8 lists the acronyms that may be used in this document.

Table 8: Acronyms

Acronym	Definition
BLMJ	Billing Major
CP	Content Provider
EVT	Event
GSM	Global System for Mobile Communication
GW	Gateway
LA	Large Account
MO	Mobile Originated (transaction)
MMSC	Multimedia Message Service Component (network)
MNP	Mobile Number Portability
MT	Mobile Terminated (transaction), Message Type
RB	Reverse Billing
SMS	Short Message Service
SMSC	Short Message Service Component (network)
SMPP	Short Message Protocol
SMTP	Short Message Transfer Protocol
TR	Transaction (as in TR_ID)
TTL	Time To Live
TTS	Time To Send

Contacting Cellact

Cellact is committed to customer service and support.

Contact us at: <mailto:support@cellact.com>

Phone: +972-9-9704181

Cellact Ltd.

Shefayim Business Center

P.O. Box 286

Shefayim 60990 Israel

Tel: +972-9-970-4110

Fax: +972-9-970-4210

Visit our website:

<http://www.cellact.com> or <http://www.cellact.co.il>