Simple Solutions Making a Big Difference

Version 7.7 Dated 03/05/2019

1. Intended Audience
Boom-SMS Software Partners and Users

2. Latest changes
- Updated URLs to use HTTPS for better security.

3. Contents

4. How it works
The following is a complete guide to enable you to connect to the Boom-SMS System
(Business to business / consumer Mobile Phone Text messaging system)
The end product will result in you configuring your software to utilise Boom-SMS for all SMS text messaging.

5. What Next
Ensure you have read the ‘Terms and Conditions’ below. Then go to www.boom-SMS.co.uk and register. Once registered you will have 10 free test SMS allocated to your account. You will receive an email confirming your account details. Should you require more test SMS contact info@boom-sms.co.uk

You will then need to integrate your software to use the Boom-SMS system. (Technical details below)
Once integration is complete and successfully tested, send out updates of your software throughout your company and/or Clients along with a short paragraph as to how the system works. Your company will then need to credit the Boom-SMS system using the 'Wordplay Secure Payment Server'. To do this the simply go to www.boom-sms.co.uk, then login and go to 'My Account' and 'Buy Credits' and choose the required amount of SMS.

Your Clients will need to ‘Register’ their own account then buy credits in the same way. Once they have entered their Boom-SMS registration details into the software and credited their account, they will be able to start sending

SMS immediately.

You can send SMS to virtually any mobile in the world provided the number is given in it's international format. For further details email: info@boom-SMS.co.uk and we will be more than happy to assist.

6. Terms and Conditions

It is at the sole discretion of the developer if he or she wishes to utilise this service and whether to present it as your own work.

Any conflicts of interest with your contract of employment/engagement will need to be dealt with by you and Boom-SMS cannot have any legal liability.

Users of this service may contact Boom-SMS, however any faults due to incorrect integration to our system will be referred back to you. We highly recommend all developers extensively test the system prior to full implementation.

Whilst every care has been taken to ensure our system will work in unison to any software it is added to, no responsibility can be taken by Boom-SMS for any disruption to that software.

Boom-SMS may change all or part of these Terms and Conditions from time to time in its sole discretion. Boom-SMS will notify its developers of any such change by email and any use of the Website or Services after such notice by such a developer will be deemed to be an acceptance of any such change. Boom-SMS will also update the Terms and Conditions contained on the Website to reflect any such change.

Boom-SMS shall use reasonable endeavours to keep the Website and Services free from viruses and corrupt files but does not warrant that neither the Website nor the Services are free from infection by viruses or anything else with contaminating or destructive properties.

All images, graphics, text and other materials on the Website are protected by copyrights owned or licensed to Boom-SMS. All trademarks, company names, software and logos on the Website, registered or unregistered are the property of Boom-SMS or have been licensed to Boom-SMS for use on the Website.

Termination: Boom-SMS may terminate your account at any time by giving you no less than 14 days written notice.

Boom-SMS may terminate your account immediately if you commit a breach of any of these Terms and Conditions.

You may terminate your account at any time by notifying Boom-SMS in writing and deleting or destroying all details provided to you. Any notice required to be given under these Terms and Conditions will be in writing and in English and must be sent by you to Boom-SMS.

English law will apply to the Terms and Conditions and any disputes will be settled in the English Courts.
We hope, like many others, that you will gain much credibility by utilising our services. Integrating to Boom-SMS means you have read, understood and agree to the above Terms and Conditions.

7. REST API

7.1 Sending SMS via HTTP

This option for sending SMS messages is best suited for platforms capable of generating an HTTP POST request. The first required step in sending SMS this way is to create a string that describes the message you want to send in XML format as described below.

```
<messages>
<sms.message>
<account.id></account.id>
<password></password>
<username></username>
<mobile.to></mobile.to>
<message></message>
<notify></notify>
<sender.email></sender.email>
</sms.message>
</messages>
```

The meaning of these data fields and what data you should provide in each of them is described below.

account.id	this is the end user's ***Username*** - obtained on registering on the Boom SMS website.
Password	this is the end user's ***Password*** - obtained on registering on the Boom SMS website
Username	*Optional*. *Max 20 Characters*. ***Personal Identifier*** - If your application stores the username of the individual originating the SMS message you can populate this field with that name. This allows more detailed reporting of who is sending messages from a multi user application
mobile.to	The ***destination Mobile Number***. May be multiple (see example 1.)
message	*Max 160 Characters*. ***The SMS message***.
notify	1 = Yes 0 = No
sender.email	*Optional*. This address is used to send notifications and other error/status messages back to the originator. If you send via email this address will override the email address contained in the email. If you send by HTTP and you do not supply this then notification messages will be sent to the email of the account holder address

Simple Solutions Making a Big Difference

Example 1 – Single message to multiple recipients:

```
<messages>
<sms.message>
<account.id>enduser</account.id>
<password>letmein</password>
<username>JOHN</username>
<mobile.to>07891234567</mobile.to>
<mobile.to>07891987654</mobile.to>
<message>Happy Birthday!!! Love Sophie</message>
<notify>1</notify>
<sender.email>someone@your-domain.com</sender.email>
</sms.message>
</messages>
```

Example 2 – Multiple messages to multiple recipients:

```
<messages>
<sms.message>
<account.id>enduser</account.id>
<password>letmein</password>
<username>JOHN</username>
<mobile.to>07891234567</mobile.to>
<mobile.to>07891987654</mobile.to>
<message>Happy Birthday!!! Love Sophie</message>
<notify>1</notify>
<sender.email>someone@your-domain.com</sender.email>
</sms.message>
<sms.message>
<account.id>enduser</account.id>
<password>apassword</password>
<username>JOHN</username>
<mobile.to>+44789123456</mobile.to>
<mobile.to>+44734567890</mobile.to>
<message>See you later</message>
<notify>1</notify>
<sender.email>someone@your-domain.com</sender.email>
</sms.message>
</messages>
```

Once you have constructed your XML you just need to send it as POST data to https://www.boom-sms.co.uk/cgi-bin/sendsms.pl

7.2 Account and Message management via HTTP

We also provide a number of options to allow users to access information regarding their account and the messages they have sent, and to perform operations such as transferring credits from one account to another. Unlike sending messages these operations do not require POST data, but rather operate based upon a query string that you provide. Below is a description of each operation, its base URL, the query string fields it supports, and an example URL query. To test the operation with the example query simply copy the query string into the end of the operations base URL and load it in a web browser.

Please note that we consider it best practice for our partners to log all messages in their database upon submission and to update these records with data pushed to them via POST. These functions are not intended to provide real-time reporting.

7.2.1 CSV Statement

Gets a list of messages this account has sent between the provided dates, formatted in CSV.
Base URL: https://services.boom-sms.co.uk/BoomPartnerASP/csv_statement/
Supported Fields: Username: the login name of your account.
Password: the login password of your account.
Sdate: the oldest date to retrive messages from, formatted as DD.MM.YYYY
Edate: the newest date to retrive messages from, formatted as DD.MM.YYYY
Example Query String:

```
?Username=testuser&Password=testpass&sdate=01.01.2010&edate=31.12.2010
```

7.2.2 XML Statement

Gets a list of messages this account has sent between the provided dates, formatted in XML.
Base URL: https://services.boom-sms.co.uk/BoomPartnerASP/xml_statement/
Supported Fields: Username: the login name of your account.
Password: the login password of your account.
Sdate: the oldest date to retrive messages from, formatted as DD.MM.YYYY
Edate: the newest date to retrive messages from, formatted as DD.MM.YYYY
Mobile.to: the MSISDN that the message was sent to.
Msgusername: the custom ID field the message was submitted with.
Status: the current status of the message (Pending, Success, Failed, etc.)
Example Query String:

```
?Username=testuser&Password=testpass&sdate=01.01.2010&edate=31.12.2010
```

7.2.3 Get Credits

Gets the number of credits currently remaining on the specified account.
Base URL: https://services.boom-sms.co.uk/BoomPartnerASP/Get_Subaccount_Credits_Int/
Supported Fields: Username: the login name of your account.
Password: the login password of your account.
SmsAccountUN: the login name of the subaccount you're retriving credits for.
Example Query String:

```
?Username=testuser&Password=testpass&SmsaccountUN=subaccount
```

7.2.4 Get Credit Transactions

Gets details of all past credit transactions.
Base URL: https://services.boom-sms.co.uk/BoomPartnerASP/Get_Credit_Transactions/
Supported Fields: Username: the login name of your account.
Password: the login password of your account.
Example Query String:

```
?Username=testuser&Password=testpass
```

7.2.5 Transfer Credits

Transfer credits from your account to another account.

Base URL: https://services.boom-sms.co.uk/BoomPartnerASP/Transfer_Credits/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.
SmsAccountUN: the login name of the subaccount you're transfering credits to.
Amount: the number of credits to move onto this account.

Example Query String:

```
?Username=testuser&Password=testpass&SmsaccountUN=subaccount&Amount=10
```

7.2.6 Change Password

Change the password of your account.

Base URL: https://services.boom-sms.co.uk/BoomPartnerASP/Account_Change_Password/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.
SmsAccountUN: the login name of the subaccount you'rechanging the password of.
OldPass: the current password of the account.
NewPass: the desired new password of the account.

Example Query String:

```
?
Username=testuser&Password=testpass&SmsaccountUN=subaccount&OldPass=notagoodpass&NewPass=agoodpass
```

7.2.7 Update Account Details

Change the details on your account.

Base URL: https://services.boom-sms.co.uk/BoomPartnerASP/Account_Update/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.
SmsAccountUN: the login name of the subaccount you're updating.
Email: A string value containing the contact email address for the account (optional field).
FirstName: A string value containing the First name of the contact (optional field).
LastName: A string value containing the Last name of the contact (optional field).
ContactTel: A string value containing the a phone number for the contact (optional field).
SenderId: A string value containing the desired SMS Sender ID to appear on text messages (optional field).
CompanyName: A string value containing the company name (optional field).
SecondaryEmail: A string value containing a secondary contact email address (optional field).
NotifyURL: A string value containing a URL that all delivery reports should be sent to via http post (optional field).

Example Query String:

```
?
Username=testuser&Password=testpass&SMSAccountUN=subaccount&firstname=Eddard&lastname=Stark&Email=someone@somwhere.com&ContactTel=07700900000&SenderId=somecorp&CompanyName=somecorp&SecondaryEmail=someoneelse@somwhereelse.co.uk&notifyURL=https://services.somecorp.co.uk/boom/
```

7.2.8 Get Account Details

Get all details stored for a given subaccount. Formatted in XML.

Base URL: https://services.boom-sms.co.uk/BoomPartnerASP/Get_Subaccount_Details/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.
SmsAccountUN: the login name of the subaccount you're retreving details of.

Example Query String:

```
?Username=testuser&Password=testpass&SmsaccountUN=subaccount
```

7.2.9 Get Account Contacts

Get all contacts stored for a given subaccount. Formatted in XML.

Base URL: https://services.boom-sms.co.uk/BoomPartnerASP/Get_Subaccount_Contacts/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.
SmsAccountUN: the login name of the subaccount you're retreving contacts of.

Example Query String:

```
?Username=testuser&Password=testpass&SmsaccountUN=subaccount
```

7.2.10 Add Contact

Add a new contact onto an account.

Base URL: https://services.boom-sms.co.uk/BoomPartnerASP/Add_Subaccount_Contact/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.
SmsAccountUN: the login name of the subaccount you're adding a contact to.
FirstName: A string containing the desired contacts first name.
Email: A string containing the desired contact email address.
Tel: A string containing the desired contact telephone number.
Type: Optional. A value of the enumerated type ContactType.
LastName: Optional. A string containing the desired contacts last name.
Mob: Optional. A string containing the desired contact mobile number.
Address: Optional. A string containing the desired contact address.
Postcode: Optional. A string containing the desired contact postal code.
NotifSent: Optional. A Boolean value indicating if this contact should receive sent message notifications.
NotifFail: Optional. A Boolean value indicating if this contact should receive failed message notifications.
NotifSent: Optional. A Boolean value indicating if this contact should receive sent message notifications.
NotifInvoice: Optional. A Boolean value indicating if this contact should receive Invoices.
NotifWP: Optional. A Boolean value indicating if this contact should receive World Pay transaction notifications.
NotifWeekly: Optional. A Boolean value indicating if this contact should receive weekly statements.
NotifLowcreds: Optional. A Boolean value indicating if this contact should receive low credits notifications.

Example Query String:

```
?
Username=testpartner&Password=partnerpasspass&SmsaccountUN=subaccount&FirstName=fname&Email
=someone@somwhere.com&Tel=01189998819991197253&LastName=lname&Mob=07000900000&Address=42aro
ad&Postcode=GU164QA&NotifSent=false&NotifFail=true&NotifReceived=false&NotifInvoice=true&No
tifWorldpay=false&NotifWeekly=true&NotifCreds=false
```

7.2.11 Remove Contact

Remove a contact for a given subaccount.

Base URL: https://services.boom-sms.co.uk/BoomPartnerASP/Remove_Subaccount_Contact/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.
SmsAccountUN: the login name of the subaccount you're removing a contact from.
ContactName: A string containing the desired contacts first name.

Example Query String:

```
?
Username=testuser&Password=testpass&SmsaccountUN=subaccount&ContactName=person
```

7.2.12 Get Subaccounts

Get a list of all subaccounts under your account formatted in XML.

Base URL: https://services.boom-sms.co.uk/BoomPartnerASP/Get_Subaccounts/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.

Example Query String:

```
?Username=testuser&Password=testpass
```

7.2.13 Get Subaccounts Latest Message

Get a list of all subaccounts under your account, and the date and time of the last message they sent, formatted in XML.

Base URL: https://services.boom-sms.co.uk/BoomPartnerASP/Get_Subaccounts_Latest_Message/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.

Example Query String:

```
?Username=testuser&Password=testpass
```

7.2.14 Get Subaccounts

Get a list of all subaccounts under your account formatted in XML.

Base URL: https://services.boom-sms.co.uk/BoomPartnerASP/Get_Inactive_Subaccounts/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.
Timeframe: A value of the timeframe enumerated type described above. Used to specify the amount of time any account should have been inactive for in order to be reported by this call.

Example Query String:

```
?Username=testuser&Password=testpass&timeframe=month
```

7.2.15 Send SMS

An alternative to sending SMS via POST as described above in section 7.1.

Base URL: https://services.boom-sms.co.uk/BoomUserASP/Smsobject_Send/

Supported Fields: account.id: the login name of your account.
password: the login password of your account.
mobile.to: the destination mobile number
message: the text of the sms message to be sent
notify: allows you to specify weather you wish to recive notifications on the success/failure of this message
sender.email: an email address to send any desired notifications to

Example Query String:

```
?account.id=testuser&password=testpass&mobile.to=07700900000&message=testing123&notify=true&sender.email=someone@somwhere.com
```

7.2.16 Send Scheduled SMS

An alternative to sending SMS via POST as described above in section 7.1. This version allows you to specify an exact date and time for the message to be delivered.

Base URL: https://services.boom-sms.co.uk/BoomUserASP/Smsobject_Send_Delayed/

Supported Fields: account.id: the login name of your account.
password: the login password of your account.
mobile.to: the destination mobile number
message: the text of the sms message to be sent
notify: allows you to specify weather you wish to recive notifications on the success/failure of this message
sender.email: an email address to send any desired notifications to

Example Query String:

```
?account.id=testuser&password=testpass&mobile.to=07700900000&message=testing123&notify=true&sender.email=someone@somwhere.com
```

7.2.17 Check Login Credentials

Check your login credentials to see if they're valid for use.

Base URL: https://services.boom-sms.co.uk/BoomUserASP/Credential_Check/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.

Example Query String:

```
?Username=testuser&Password=testpass
```

7.2.18 Check Credits

Gets the total number of credits remaining on this account.

Base URL: https://services.boom-sms.co.uk/BoomUserASP/Credit_Check/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.

Example Query String:

```
?Username=testuser&Password=testpass
```

7.2.19 View Invoices

Get all invoices stored against your account. Formatted in XML.

Base URL: https://services.boom-sms.co.uk/BoomUserASP/User_Invoices_XML/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.

Example Query String:

```
?Username=testuser&Password=testpass
```

7.2.20 View Unpaid Invoices

Get all outstanding invoices stored against your account. Formatted in XML.

Base URL: https://services.boom-sms.co.uk/BoomUserASP/User_unpaid_Invoices_XML/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.

Example Query String:

```
?Username=testuser&Password=testpass
```

7.2.21 View Account Details

View all stored details for the current account. Formatted in XML.

Base URL: https://services.boom-sms.co.uk/BoomUserASP/Account_Details_XML/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.

Example Query String:

```
?Username=testuser&Password=testpass
```

7.2.22 View Sent Messages

View Messages that this account has sent. Formatted in XML.

Base URL: https://services.boom-sms.co.uk/BoomUserASP/User_Messages_XML/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.

Example Query String:

```
?Username=testuser&Password=testpass
```

7.2.23 View Account Contacts

View Contacts for the current account. Formatted in XML.

Base URL: https://services.boom-sms.co.uk/BoomUserASP/User_Contacts_XML/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.

Example Query String:

```
?Username=testuser&Password=testpass
```

7.2.24 Remove Contact

View Contacts for the current account. Formatted in XML.

Base URL: https://services.boom-sms.co.uk/BoomUserASP/Remove_Contact/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.
ContactName: A string containing the desired contacts first name.

Example Query String:

```
?Username=testuser&Password=testpass&ContactName=person
```

7.2.25 Add Contact

Add a new contact onto the current account.

Base URL: https://services.boom-sms.co.uk/BoomUserASP/Add_Contact/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.
FirstName: A string containing the desired contacts first name.
Email: A string containing the desired contact email address.
Tel: A string containing the desired contact telephone number.
Type: Optional. A value of the enumerated type ContactType.
LastName: Optional. A string containing the desired contacts last name.
Mob: Optional. A string containing the desired contact mobile number.
Address: Optional. A string containing the desired contact address.
Postcode: Optional. A string containing the desired contact postal code.
NotifSent: Optional. A Boolean value indicating if this contact should receive sent message notifications.
NotifFail: Optional. A Boolean value indicating if this contact should receive failed message notifications.
NotifSent: Optional. A Boolean value indicating if this contact should receive sent message notifications.
NotifInvoice: Optional. A Boolean value indicating if this contact should receive Invoices.
NotifWP: Optional. A Boolean value indicating if this contact should receive World Pay transaction notifications.
NotifWeekly: Optional. A Boolean value indicating if this contact should receive weekly statements.
NotifLowcreds: Optional. A Boolean value indicating if this contact should receive low credits notifications.

Example Query String:

```
?
Username=testpartner&Password=partnerpasspass&FirstName=fname&Email=someone@somwhere.com&Tel=01189998819991197253&LastName=lname&Mob=07000900000&Address=42aroad&Postcode=GU164QA&NotifSent=false&NotifFail=true&NotifReceived=false&NotifInvoice=true&NotifWorldpay=false&NotifWeekly=true&NotifCreds=false
```

7.2.26 View Templates

View all message templates for the current account. Formatted in XML.

Base URL: https://services.boom-sms.co.uk/BoomUserASP/User_Templates_XML/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.

Example Query String:

```
?Username=testuser&Password=testpass
```

7.2.27 Add Template

Add a message template for the current account.

Base URL: https://services.boom-sms.co.uk/BoomUserASP/Add_Template/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.
Template: the desired message that you wish to save as a template.

Example Query String:

```
?Username=testuser&Password=testpass&Template=somemessage
```

7.2.28 Remove Template

Remove a message template for the current account.

Base URL: https://services.boom-sms.co.uk/BoomUserASP/Remove_Template/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.
Template: the message template you wish to remove from the account.

Example Query String:

```
?Username=testuser&Password=testpass&Template=somemessage
```

7.2.29 Update Account Details

Change the details on your account.

Base URL: https://services.boom-sms.co.uk/BoomUserASP/Account_Update/

Supported Fields: Username: the login name of your account.
Password: the login password of your account.
NewPassword: A string value containing hte updated password you want on your account (optional field).
Email: A string value containing the contact email address for the account (optional field).
FirstName: A string value containing the First name of the contact (optional field).
LastName: A string value containing the Last name of the contact (optional field).
ContactTel: A string value containing the a phone number for the contact (optional field).
SenderId: A string value containing the desired SMS Sender ID to appear on text messages (optional field).
CompanyName: A string value containing the company name (optional field).
SecondaryEmail: A string value containing a secondary contact email address (optional field).
NotifyURL: A string value containing a URL that all delivery reports should be sent to via http post (optional field).

Example Query String:

```
?
Username=testuser&Password=testpass&NewPassword=newpass&firstname=Jane&lastname=Doe&Email=someone@somwhere.com&ContactTel=07700900000&SenderId=somecorp&CompanyName=somecorp&SecondaryEmail=someoneelse@somwhereelse.co.uk&notifyURL=https://services.somecorp.co.uk/boom/
```

7.3 Receiving SMS

If you have requested to be able to receive SMS with us then we will have assigned you a *Virtual Number* to distribute to your customers. This number is assigned to you and only you and any SMS sent to it can be communicated to you in two ways.

7.3.1 Email

Once you have logged into our website and navigated to My Account > Edit Settings then under Notification & Received Message Settings there is an option titled *Receive Email.* The email address you enter here will be notified every time an SMS is sent to your Virtual Number.

You can also set specific contacts to receive these emails as well by navigating to My Account > Contacts and then clicking Edit for the contact you wish to receive email notifications. On the next screen under *E-mail Notifications* check *Notify Received Messages* to enable these notifications for this contact.

7.3.2 HTTP POST

Once you have logged into our website and navigated to My Account > Edit Settings then under Notification & Received Message Settings there is an option titled *Receive URL.* We will send POST data to this URL every time an SMS is sent to your virtual number. An example of the kind of POST data you can expect to receive is below

```
MessageDate=26%2f06%2f2014+16%3a00%3a54&MobileFrom=447860020078&Message=This+is+a+test+message
```

7.4 Notifications

Once you have sent an SMS you can check on its status by navigating to My Account > Reports > Sent Messages. You can also choose to be informed of its status automatically in two ways.

7.4.1 Email

Once you have logged into our website and navigated to My Account > Contacts you can then click Edit for the contact you wish to receive email notifications. On this screen under *E-mail Notifications* you can check what kind of notifications you wish this contact to receive. "Notify Sms Message Sent" will cause an email to be sent to this person every time a message is successfull. "Notify Sms Message Failed" will cause an email to be sent to this person every time a message fails.

7.4.2 HTTP POST

Once you have logged into our website and navigated to My Account > Edit Settings then under Notification & Received Message Settings there is an option titled *Notify URL.* We will send POST data to this URL every time an SMS is statused. An example of the kind of POST data you can expect to receive is below

```
AccountName=testuser&DateSent=26%2f06%2f2014+16%3a00%3a54&User=Test+User+&MobileTo=447860020078&StatusDate=26%2f06%2f2014+16%3a01%3a04&Status=Success&PTTCode=4&PTTDescription=Delivered+Message+-
+Your+message+has+been+delivered+to+the+recipients+device.&PTTType=4&PTTTypeDescription=Delivered
```

8. SOAP API

This is a WCF API that is intended to be implemented in an environment using the Microsoft .net architecture. However since it uses a SOAP transport it can be implemented in any environment capable of generating a SOAP request.

This API exposes some custom data types in addition to functions that are designed to be used in combination. The details of both are in the next section. At the end of this document we have provided some example code in the C# language to aid you in integrating this with your software.

We also recommend that users seeking to evaluate our API before integrating with their own software do so with the excellent free testing tool soapUI, made by eviware and available from soapui.org.

8.1 API Locations

This API is split into User and Partner sections, each encompassing functionality intended for use by either Users or software Partners. Each of these sections is then further broken down into sub-sections that group together API calls of similar purpose. In the section below each of these sections is laid out and the API calls they contain are described.

The base URL for each of these are below. If you require WSDL data then add "?wsdl" to the end of these URLs.

Partner Base URL: https://services.boom-sms.co.uk/accountservices/boompartner.svc
User Base URL: https://services.boom-sms.co.uk/accountservices/boomuser.svc

8.2 Partner Functionality:

8.2.1 Custom Data Types

8.2.1.1 Enumerated type: Account_Operation_Result

Values

0. Success: Indicates a successful account creation
1. InvalidPartner: Invalid partner details
2. InvalidAuthentication: Partner is not authorised to set-up accounts remotely
3. InvalidField: Field name was not recognised
4. InvalidDateField: A supplied Date value is not in a valid date format
5. FieldNotUnique: A supplied value already exists in the database (e.g. **AccountID**)
6. EmptyValue: Indicates an essential field was not supplied with a value or was supplied with an empty value.
7. InvalidEmail: Indicates a supplied email address was not in a proper email format
8. InvalidCharacters: Indicates an invalid was supplied in a field (e.g. | , # %)
9. InternalError: Indicates an internal server error, please re-try and notify support if problem re-occurs.
10. PasswordMismatch: Indicates that when attempting to change an account password the supplied existing password did not match the existing password of the account.

8.2.1.2 Enumerated type: ContactType

Values

2. Accounts: Specifies that this contact is an Accounts contact.
3. Technical: Specifies that this contact is a Technical contact.
4. Partner: Specifies that this contact is a Partner contact.
6. Other: Specifies that this contact is another kind contact.

8.2.1.3 Enumerated type: TimeFrame

Values

1. Day: Specifies a span of time equal to 24 hours.
2. Week: Specifies a span of time equal to 7 days.
3. Month: Specifies a span of time equal to 30 days.
4. Year: Specifies a span of time equal to 365 days.

8.2.1.4 Class: Statement_item

Values

strUsername: a string containing the username of the user the message was sent to
strMobileno: a string containing the mobile number the message was sent to
strUser_ID: a string containing the ID of the user the message was sent to
strSmsMessage: a string containing the contents of the message that was sent
strCreditsUsed: a string containing the number of credits the message used
datCreatedOn: DateTime variable used to indicate the date message was sent
datUpdateOn: a DateTime variable used to indicate the message was updated
strNotificationStatus: a string used to indicate the status of the message

8.2.1.5 Class: searchcriteria

Note: all fields are optional, specify only desired values

Values

acusername: a string containing the username of the user
msgusername: a string containing the message identifier
mobto: a string containing the destination mobile number
msg: a string containing the contents of the message that was sent
dstatus: object of class delivery_status, possible values are 1 (pending), 2 (failed), 3 (succeeded), 4 (blank), 5(unknown) and 100(all). Please note that although this class is used is many API calls the **dstatus** element is only relevant when used with the Statements functions.

8.2.1.6 Class: Partner_credentials

Values

partnerun: a string containing the username of the partner
partnerpw: a string containing the password of the user

The Methods of this API are split into two interfaces, one encapsulating Account_Management functionality, and another encapsulating account statements functionality.

8.2.2 Interface: Statements

Please note that we consider it best practice for our partners to log all messages in their database upon submission and to update these records with data pushed to them via POST. These functions are not intended to provide real-time reporting.

Function: csv_statement

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.
search_criteria: an object of class searchcriteria, used to specify terms to search for.
sdate: a DateTime variable, used to indicate oldest messages to search.
edate: a DateTime variable, used to indicate newest messages to search.

Return

CSV_Data: a string containing search results in a comma delimited format.

Function: dataset_statement_date

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.
search_criteria: an object of class searchcriteria, used to specify terms to search for.
sdate: a DateTime variable, used to indicate oldest messages to search.
edate: a DateTime variable, used to indicate newest messages to search.

Return

edataset: a dataset containing search result.

Function: dataset_statement

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.
Search_criteria: an object of class searchcriteria, used to specify terms to search for.

Return

edataset: a dataset containing search results

Function: collection_statement

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.
search_criteria: an object of class searchcriteria, used to specify terms to search for.
sdate: a DateTime variable, used to indicate oldest messages to search.
edate: a DateTime variable, used to indicate newest messages to search.

Return

DataStatements: a list of objects of the class statement_item each containing one set of search results

Function: XML_statement

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.
search_criteria: an object of class searchcriteria, used to specify terms to search for.
sdate: a DateTime variable, used to indicate oldest messages to search.
edate: a DateTime variable, used to indicate newest messages to search.

Return

s: a string containing search results as XML formatted data.

Function: Get_Subaccount_Credits_Dataset

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.
SmsaccountUN: A string value containing the name of the subaccount to retrieve the current credit total of. This field can contain wildcard characters such as % to match any string or _ to match any single character.

Return

T: a system.data.dataset object containing the Sms Account name and total credits. Will return an empty dataset if either **SmsaccountUN** or **SmsaccountPW** were incorrect.

Function: Get_Subaccount_Credits_Int

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.
SmsaccountUN: A string value containing the name of the subaccount to retrieve the current credit total of

Return

I: an integer variable containing the total credits for the indicated account. Will return a negative value if **SmsaccountUN** was incorrect.

Function: Get_Credit_Transactions_Dataset

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

Return

T: a system.data.dataset object containing details of all past credit transactions. Will return an empty dataset if partner details were incorrect.

Function: Get_Credit_Transactions_XML

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

Return

S: a string containing details of all past credit transactions. Will return an empty dataset if partner details were incorrect.

Function: Dataset_Failed_Messages_Date

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

search_criteria: an object of class searchcriteria, used to specify terms to search for.

sdate: a DateTime variable, used to indicate oldest messages to search.

edate: a DateTime variable, used to indicate newest messages to search.

Return

edataset: a dataset containing details of messages that have been submitted but failed to send.

Function: Dataset_Failed_Messages

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

search_criteria: an object of class searchcriteria, used to specify terms to search for.

Return

edataset: a dataset containing details of messages that have been submitted but failed to send.

Function: XML_Failed_Messages_Date

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

search_criteria: an object of class searchcriteria, used to specify terms to search for.

sdate: a DateTime variable, used to indicate oldest messages to search.

edate: a DateTime variable, used to indicate newest messages to search.

Return

s: a string containing details of messages that have been submitted but failed to send as XML formatted data.

Function: XML_Failed_Messages

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

search_criteria: an object of class searchcriteria, used to specify terms to search for.

Return

s: a string containing details of messages that have been submitted but failed to send as XML formatted data.

8.2.3 Interface: Account_Management

Function: Account_Create

Note: use a partner name of TEST to perform a dry run, return value will be accurate for the data provided but no account will be created.

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.
Username: A string value containing the desired username for the account
Password: A string value containing the desired Password for the account
Email: A string value containing the contact email address for the account
ContactFirstName: A string value containing the First name of the contact
ContactLastName: A string value containing the Last name of the contact
ContactTel: A string value containing the a phone number for the contact
SMSSenderId: A string value containing the desired SMS Sender ID to appear on text messages (optional field).
CompanyName: A string value containing the company name (optional field).
SecondaryEmail: A string value containing a secondary contact email address (optional field).
NotifyURL: A string value containing a URL that all delivery reports should be sent to (optional field).

Return

Account_Create_Result: a value of the enumerated type Account_Create_result (see above for specifics).

Function: Transfer_Credits

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.
SmsaccountUN: A string value containing the name of the subaccount to add credits too
Amount: An integer value indicating the number of credits to be transferred from the partner account to the user account

Return

b: A Boolean value with true indicating the credit transfer was successful and false indicating a failure

Function: Account_Change_Password

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.
SmsaccountUN: A string value containing the name of the subaccount to change the password of.
OldPass: A string value containing the current password of the account.
NewPass: A string value containing the desired new password of the account.

Return

b: A Boolean value with true indicating the password change was successful and false indicating a failure

Function: Account_Update

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.
Username: A string value indicating the username of the account to update
Email: A string value containing the contact email address for the account (optional field).
ContactFirstName: A string value containing the First name of the contact (optional field).
ContactLastName: A string value containing the Last name of the contact (optional field).
ContactTel: A string value containing the a phone number for the contact (optional field).
SMSSenderId: A string value containing the desired SMS Sender ID to appear on text messages (optional field).
CompanyName: A string value containing the company name (optional field).
SecondaryEmail: A string value containing a secondary contact email address (optional field).
NotifyURL: A string value containing a URL that all delivery reports should be sent to via http post (optional field).

Return

Account_Update_Result: a value of the enumerated type Account_Create_result (see above for specifics).

Function: Get_Subaccount_Details_Dataset

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.
Username: A string value indicating the username of the account to retrieve details for

Return

T: a system.data.dataset object containing all stored details for the specified account. Will return an empty dataset if any credentials are incorrect.

Function: Get_Subaccount_Details_XML

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.
Username: A string value indicating the username of the account to retrieve details for

Return

S: a string variable containing all stored details for the specified account formatted in an XML layout. Will return an empty dataset if any credentials are incorrect.

Function: Get_Subaccount_Contacts_Dataset

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

ContactName: A string containing the username of the target subaccount.

Return

RetData: A system.data.dataset variable that contains all stored information regarding this contact.

Function: Get_Subaccount_Contacts_XML

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

ContactName: A string containing the username of the target subaccount.

Return

RetData: A string value containing all stored information regarding your contacts formatted in XML.

Function: Add_Subaccount_Contact

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

Username: A string containing the username of the target subaccount.

FirstName: A string containing the desired contacts first name.

Email: A string containing the desired contact email address.

Tel: A string containing the desired contact telephone number.

Type: Optional. A value of the enumerated type ContactType.

LastName: Optional**.** A string containing the desired contacts last name.

Mob: Optional**.** A string containing the desired contact mobile number.

Address: Optional**.** A string containing the desired contact address.

Postcode: Optional**.** A string containing the desired contact postal code.

NotifySent: Optional. A Boolean value indicating if this contact should receive sent message notifications.

NotifyFail: Optional. A Boolean value indicating if this contact should receive failed message notifications.

NotifySent: Optional. A Boolean value indicating if this contact should receive sent message notifications.

NotifyInvoice: Optional. A Boolean value indicating if this contact should receive Invoices.

NotifyWP: Optional. A Boolean value indicating if this contact should receive World Pay transaction notifications.

NotifyWeekly: Optional. A Boolean value indicating if this contact should receive weekly statements.

NotifyLowcreds: Optional. A Boolean value indicating if this contact should receive low credits notifications.

Return

b: A Boolean value indicating success or failure in adding this contact.

Function: Remove_Subaccount_Contact

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

Username: A string containing the username of the target subaccount.

ContactName: A string containing the desired contacts first name.

Return

Contact_Create_Result: a value of the enumerated type Account_Operation_result (see above for specifics).

Function: Get_Subaccounts_Dataset

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

Return

RetData: A system.data.dataset variable that contains all stored information regarding your sub-accounts.

Function: Get_Subaccounts_XML

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

Return

RetData: A string value containing all stored information regarding your sub-accounts formatted in XML.

Function: Get_Subaccounts_Latest_Messages_Dataset

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

Return

RetData: A system.data.dataset variable that contains the username of each of your sub-accounts and the date and time of the last message they sent.

Function: Get_Subaccounts_Latest_Messages_XML

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

Return

RetData: A string value containing the username of each of your sub-accounts and the date and time of the last message they sent, formatted in XML.

Function: Get_Inactive_Subaccounts_Dataset

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

Timeframe: A value of the timeframe enumerated type described above. Used to specify the amount of time any account should have been inactive for in order to be reported by this call.

Return

RetData: A system.data.dataset variable that contains the username of each of your sub-accounts and the date and time of the last message they sent.

Function: Get_Inactive_Subaccounts_XML

Parameters

partner_account: An object of class Partner_credentials, used to identify the partner account to be used.

Timeframe: A value of the timeframe enumerated type described above. Used to specify the amount of time any account should have been inactive for in order to be reported by this call.

Return

RetData: A string value containing the username of each of your sub-accounts and the date and time of the last message they sent, formatted in XML.

8.3 User Functionality:

8.3.1 Custom Data Types

8.3.1.1 Enumerated type: Account_Operation_Result

Values

0. Success: Indicates a successful account creation
1. InvalidPartner: Invalid partner details
2. InvalidAuthentication: Partner is not authorised to set-up accounts remotely
3. InvalidField: Field name was not recognised
4. InvalidDateField: A supplied Date value is not in a valid date format
5. FieldNotUnique: A supplied value already exists in the database (e.g. AccountID)
6. EmptyValue: Indicates an essential field was not supplied with a value or was supplied with an empty value.
7. InvalidEmail: Indicates a supplied email address was not in a proper email format
8. InvalidCharacters: Indicates an invalid was supplied in a field (e.g. | , # %)
9. InternalError: Indicates an internal server error, please re-try and notify support if problem re-occurs.
10. PasswordMismatch: Indicates that when attempting to change an account password the supplied existing password did not match the existing password of the account.

8.3.1.2 Enumerated type: ContactType

Values

2. Accounts: Specifies that this contact is an Accounts contact.
3. Technical: Specifies that this contact is a Technical contact.
4. Partner: Specifies that this contact is a Partner contact.
6. Other: Specifies that this contact is another kind contact.

8.3.1.3 Enumerated type: CredCheck

Values

0. noMatch: Indicates the specified username was not found.
1. partMatch: Indicates the username was found but the supplied password was incorrect.
2. fullMatch: Indicates the username was found and the supplied password was correct.

8.3.1.4 Class: searchcriteria

Note: all fields are optional, specify only desired values

Values

mobto: a string containing the destination mobile number
msg: a string containing the contents of the message that was sent
dstatus: value of enum delivery_status, possible values are 1 (pending), 2 (failed), 3 (succeeded), 4 (blank), 5(unknown) and 100(all).

8.3.1.5 Class: Smsclass

Values

strAccount_ID: A string containing your account ID.
strPassword: A string containing your account password.
strUsername: A string containing your account username.
strMobTo: An array of strings containing the destination mobile numbers, these can be any valid mobile number in international format or a valid UK landline number.
strMessage: A string containing the body of the message.
bNotify: A Boolean variable containing true or false to indicate notification status (optional).
strSenderEmail: A string containing the e-mail address of the sender.

The Methods of this API are split into several interfaces, one encapsulating Sending functionality, another encapsulating Checking functionality and another encapsulating retrieving message and account information.

8.3.2 Interface: Sending

Function: XML_String_Send

Parameters

post_string: a string that contains the XML data to be sent.

Return

bFound: a Boolean value that indicates if a valid <sms.message> tag was found.

Function: SMS_Send_Object

Parameters

XMLobj: an array of objects of the smsclass class.

Return

bResult: a Boolean value that indicates success or failure. This return value indicates only the receipt of your message by our server; this is **not** a final delivery status.

Function: XML_Send_String_Delayed

Parameters

post_string: a string that contains the XML data to be sent
TimeToSend: a DateTime variable that indicates the time that the message should be sent (in GMT).

Return

bFound: a Boolean value that indicates if a valid <sms.message> tag was found.

Function: SMS_Send_Object_Delayed

Parameters

XMLobj: an array of objects of the smsclass class.

TimeToSend: a DateTime variable that indicates the time that the message should be sent (in GMT).

Return

bResult: a Boolean value that indicates success or failure. This return value indicates only the receipt of your message by our server; this is **not** a final delivery status.

Function: SMS_Send_Object_Delayed_Split

Parameters

XMLobj: an array of objects of the smsclass class.

TimeToSend: An array of DateTime variables indicating the times you would like this batch to be sent. The supplied messages will be split evenly into batches to be sent on each of the given dates.

Return

bResult: a Boolean value that indicates success or failure. This return value indicates only the receipt of your message by our server; this is **not** a final delivery status.

8.3.3 Interface: Checking

Function: Credential_Check

Parameters

UsrName: A string containing your account name.

Passwd: A string containing your account password.

Return

bResult: A value of the enumerated type CredCheck (see above for specifics).

Function: Credit_Check

Parameters

UsrName: A string containing your account name.

Passwd: A string containing your account password.

Return

bResult: An integer value containing the number of credits on the account.

Function: Get_Invoices_Dataset

Parameters

UsrName: A string containing your account name.

Passwd: A string containing your account password.

Return

RetData: A system.data.dataset variable that contains all invoices stored against your account.

Function: Get_Invoices_XML

Parameters

UsrName: A string containing your account name.

Passwd: A string containing your account password.

Return

RetData: An XML formatted string variable that contains all invoices stored against your account.

Function: Get_Unpaid_Invoices_Dataset

Parameters

UsrName: A string containing your account name.

Passwd: A string containing your account password.

Return

RetData: A system.data.dataset variable that contains all unpaid invoices stored against your account.

Function: Get_Unpaid_Invoices_XML

Parameters

UsrName: A string containing your account name.

Passwd: A string containing your account password.

Return

RetData: An XML formatted string variable that contains all unpaid invoices stored against your account.

Function: Get_Corrected_Numbers

Parameters

Numbers: A one dimensional string array containing the numbers to be corrected.

Return

RetData: An XML formatted string variable that contains Details of the corrected numbers.

8.3.4 Interface: Details

Note: The Get_Messages functions can time-out if being used to query a large amount of data. We therefore recommend breaking large queries down into smaller chunks by date.

Function: Get_Account_Details_Dataset

Parameters

UsrName: A string containing your account name.

Passwd: A string containing your account password.

Return

RetData: A system.data.dataset variable that contains all stored information regarding your account.

Function: Account_Update

UsrName: A string containing your account name.
Passwd: A string containing your account password.
NewPassword: A string value containing the contact email address for the account (optional field).
Email: A string value containing the contact email address for the account (optional field).
ContactFirstName: A string value containing the First name of the contact (optional field).
ContactLastName: A string value containing the Last name of the contact (optional field).
ContactTel: A string value containing the a phone number for the contact (optional field).
SMSSenderId: A string value containing the desired SMS Sender ID to appear on text messages (optional field).
CompanyName: A string value containing the company name (optional field).
SecondaryEmail: A string value containing a secondary contact email address (optional field).
NotifyURL: A string value containing a URL that all delivery reports should be sent to via http post (optional field).

Return

Account_Update_Result: a value of the enumerated type Account_Create_result (see above for specifics).

Function: Get_Account_Details_XML

Parameters

UsrName: A string containing your account name.
Passwd: A string containing your account password.

Return

RetData: A string value containing all stored information regarding your account formatted in XML.

Function: Get_Messages_Dataset

Parameters

UsrName: A string containing your account name.
Passwd: A string containing your account password.
SearchCriteria: an object of the searchcriteria class described above.
Sdate: a DateTime variable, used to indicate oldest messages to search.
Edate: a DateTime variable, used to indicate newest messages to search.

Return

RetData: A system.data.dataset variable that contains your search results

Function: Get_Messages_XML

Parameters

UsrName: A string containing your account name.
Passwd: A string containing your account password.
SearchCriteria: an object of the searchcriteria class described above.
Sdate: a DateTime variable, used to indicate oldest messages to search.
Edate: a DateTime variable, used to indicate newest messages to search.

Return

RetData: A string value that contains your search results formatted in XML.

Function: Get_Contacts_Dataset

Parameters

UsrName: A string containing your account name.
Passwd: A string containing your account password.

Return

RetData: A system.data.dataset variable that contains all stored information regarding your contacts.

Function: Get_Contacts_XML

Parameters

UsrName: A string containing your account name.
Passwd: A string containing your account password.

Return

RetData: A string value containing all stored information regarding your contacts formatted in XML.

Function: Remove_Contact

Parameters

Username: A string containing your account name.
Password: A string containing your account password.
ContactName: A string containing the desired contacts first name.

Return

Contact_Create_Result: a value of the enumerated type Account_Operation_result (see above for specifics).

Function: Add_Contact

Parameters

Username: A string containing your account name.
Password: A string containing your account password.
FirstName: A string containing the desired contacts first name.
Email: A string containing the desired contact email address.
Tel: A string containing the desired contact telephone number.
Type: Optional. A value of the enumerated type ContactType.
LastName: Optional**.** A string containing the desired contacts last name.
Mob: Optional**.** A string containing the desired contact mobile number.
Address: Optional**.** A string containing the desired contact address.
Postcode: Optional**.** A string containing the desired contact postal code.
NotifySent: Optional. A Boolean value indicating if this contact should receive sent message notifications.
NotifyFail: Optional. A Boolean value indicating if this contact should receive failed message notifications.
NotifyReceived: Optional. A Boolean value indicating if this contact should receive message notifications for received messages.
NotifyInvoice: Optional. A Boolean value indicating if this contact should receive Invoices.
NotifyWP: Optional. A Boolean value indicating if this contact should receive World Pay transaction notifications.
NotifyWeekly: Optional. A Boolean value indicating if this contact should receive weekly statements.
NotifyLowcreds: Optional. A Boolean value indicating if this contact should receive low credits notifications.

Return

Account_Operation_Result: a value of the enumerated type Account_Operation_result (see above for specifics).

Function: Get_Templates_Dataset

Parameters

UsrName: A string containing your account name.
Passwd: A string containing your account password.

Return

RetData: A system.data.dataset variable that contains all your stored message templates.

Function: Get_Templates_XML

Parameters

UsrName: A string containing your account name.
Passwd: A string containing your account password.

Return

RetData: An XML formatted string variable that contains all your stored message templates.

Function: Add_Template

Parameters

Username: A string containing your account name.
Password: A string containing your account password.
Template: A string containing the desired message template.

Return

Account_Operation_Result: a value of the enumerated type Account_Operation_result (see above for specifics).

Function: Remove_Template

Parameters

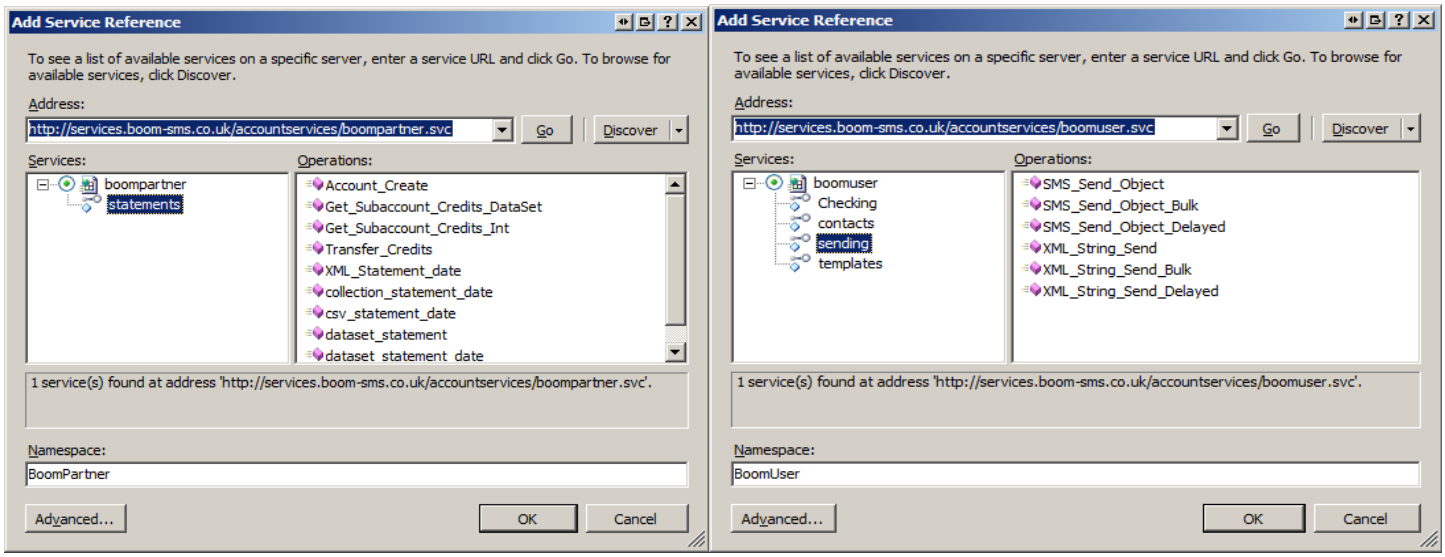
Username: A string containing your account name.
Password: A string containing your account password.
Template: A string that matches exactly the desired template to remove.

Return

Account_Operation_Result: a value of the enumerated type Account_Operation_result (see above for specifics).

8.4 Example Code

On the next page is example code written in C# using the .Net architecture in order to ilustrate how to impliment the Boom-SMS API into your own programs. All code is confirmed to work with the Boom-SMS API as of the date of this document. All clients must first register the boom-sms service reference in their project as below.



```
private void btnSubmit_CSV_Statement_Click(object sender, EventArgs e)
        {
            BoomPartner.StatementsClient svc = new BoomPartner.StatementsClient("WCFStatements");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtCSV_Statement_PartnerUN.Text;
            PartnerInfo.Partner_password = txtCSV_Statement_PartnerPW.Text;

            DateTime StartDate;
            StartDate = new DateTime();
            StartDate = DateTime.ParseExact(txtCSV_Statement_sDate.Text, "dd/MM/yyyy HH:mm:ss",null);

            DateTime EndDate;
            EndDate = new DateTime();
            EndDate = DateTime.ParseExact(txtCSV_Statement_eDate.Text, "dd/MM/yyyy HH:mm:ss",null);

            BoomPartner.searchcriteria SearchValues = new BoomPartner.searchcriteria();
            SearchValues.account_id = txtCSV_Statement_AccUsername.Text;
            SearchValues.message = txtCSV_Statement_Msg.Text;
            SearchValues.mobile_to = txtCSV_Statement_MobTo.Text;
            SearchValues.username = txtCSV_Statement_MsgUsername.Text;

            switch (cmbCSV_Statement_dStatus.SelectedText.ToLower())
            {
                case "blank":
                    SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.blank;
                    break;
                case "failed":
                    SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.failed;
                    break;
                case "pending":
                    SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.pending;
                    break;
                case "success":
                    SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.success;
                    break;
                case "unknown":
                    SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.unknown;
                    break;
                default:
                    SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.all;
                    break;
            }
```

```
    txtCSV_Statement_return.Text = svc.csv_statement_date(PartnerInfo, SearchValues, StartDate, EndDate);

}

private void btnSubmit_Dataset_Statement_Date_Click(object sender, EventArgs e)
{
    BoomPartner.StatementsClient svc = new BoomPartner.StatementsClient("WCFStatements");
    BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
    PartnerInfo.Partner_username = txtDataset_Statement_Date_PartnerUN.Text;
    PartnerInfo.Partner_password = txtDataset_Statement_Date_PartnerPW.Text;

    DateTime StartDate;
    StartDate = new DateTime();
    StartDate = DateTime.ParseExact(txtDataset_Statement_Date_Sdate.Text, "dd/MM/yy HH:mm:ss ", null);

    DateTime EndDate;
    EndDate = new DateTime();
    EndDate = DateTime.ParseExact(txtDataset_Statement_Date_Edate.Text, "dd/MM/yy HH:mm:ss ", null);

    BoomPartner.searchcriteria SearchValues = new BoomPartner.searchcriteria();
    SearchValues.account_id = txtDataset_Statement_Date_AccUsername.Text;
    SearchValues.message = txtDataset_Statement_Date_Msg.Text;
    SearchValues.mobile_to = txtDataset_Statement_Date_MobTo.Text;
    SearchValues.username = txtDataset_Statement_Date_MsgUsername.Text;

    switch (cmbDataset_Statement_Date_Dstatus.SelectedText.ToLower())
    {
        case "blank":
            SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.blank;
            break;
        case "failed":
            SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.failed;
            break;
        case "pending":
            SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.pending;
            break;
        case "success":
            SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.success;
            break;
        case "unknown":
            SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.unknown;
```

```
                    break;
                default:
                    SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.all;
                    break;
            }

            dgvDataset_Statement_Date_return.DataSource = svc.dataset_statement_date(PartnerInfo, SearchValues,
StartDate, EndDate);
        }

        private void btnSubmit_Dataset_Statement_Click(object sender, EventArgs e)
        {
            BoomPartner.StatementsClient svc = new BoomPartner.StatementsClient("WCFStatements");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtDataset_Statement_PartnerUN.Text;
            PartnerInfo.Partner_password = txtDataset_Statement_PartnerPW.Text;

            BoomPartner.searchcriteria SearchValues = new BoomPartner.searchcriteria();
            SearchValues.account_id = txtDataset_Statement_AccUsername.Text;
            SearchValues.message = txtDataset_Statement_Msg.Text;
            SearchValues.mobile_to = txtDataset_Statement_MobTo.Text;
            SearchValues.username = txtDataset_Statement_MsgUsername.Text;

            switch (cmbDataset_Statement_Dstatus.SelectedText.ToLower())
            {
                case "blank":
                    SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.blank;
                    break;
                case "failed":
                    SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.failed;
                    break;
                case "pending":
                    SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.pending;
                    break;
                case "success":
                    SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.success;
                    break;
                case "unknown":
                    SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.unknown;
                    break;
                default:
                    SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.all;
```

```
            break;
        }

        dgvDataset_Statement_return.DataSource = svc.dataset_statement(PartnerInfo, SearchValues);
    }

    private void btnSubmit_Collection_Statement_Click(object sender, EventArgs e)
    {
        BoomPartner.StatementsClient svc = new BoomPartner.StatementsClient("WCFStatements");
        BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
        PartnerInfo.Partner_username = txtCollection_Statement_PartnerUN.Text;
        PartnerInfo.Partner_password = txtCollection_Statement_PartnerPW.Text;

        DateTime StartDate;
        StartDate = new DateTime();
        StartDate = DateTime.ParseExact(txtCollection_Statement_Sdate.Text, "dd/MM/yy HH:mm:ss ", null);

        DateTime EndDate;
        EndDate = new DateTime();
        EndDate = DateTime.ParseExact(txtCollection_Statement_Edate.Text, "dd/MM/yy HH:mm:ss ", null);

        BoomPartner.searchcriteria SearchValues = new BoomPartner.searchcriteria();
        SearchValues.account_id = txtCollection_Statement_AccUsername.Text;
        SearchValues.message = txtCollection_Statement_Msg.Text;
        SearchValues.mobile_to = txtCollection_Statement_MobTo.Text;
        SearchValues.username = txtCollection_Statement_MsgUsername.Text;

        switch (cmbCollection_Statement_Dstatus.SelectedText.ToLower())
        {
            case "blank":
                SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.blank;
                break;
            case "failed":
                SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.failed;
                break;
            case "pending":
                SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.pending;
                break;
            case "success":
                SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.success;
                break;
            case "unknown":
```

```
                SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.unknown;
                break;
            default:
                SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.all;
                break;
        }

        dlvCollection_Statement_return.GridLines = true;
        dlvCollection_Statement_return.Columns.Add("strUsername");
        dlvCollection_Statement_return.Columns.Add("strMobileno");
        dlvCollection_Statement_return.Columns.Add("strUser_ID");
        dlvCollection_Statement_return.Columns.Add("strSmsMessage");
        dlvCollection_Statement_return.Columns.Add("strCreditsUsed");
        dlvCollection_Statement_return.Columns.Add("strCreatedOn");
        dlvCollection_Statement_return.Columns.Add("strUpdatedOn");
        dlvCollection_Statement_return.Columns.Add("strNotificationStatus");

        BoomPartner.statement_item[] results;
        results = svc.collection_statement_date(PartnerInfo, SearchValues,StartDate,EndDate);

        foreach (BoomPartner.statement_item result in results)
        {
            string[] arr = new string[8];
            arr[0] = result.Username;
            arr[1] = result.MobileNo;
            arr[2] = result.UserID;
            arr[3] = result.SmsMessage;
            arr[4] = result.CreditsUsed;
            arr[5] = result.CreatedOn.ToString();
            arr[6] = result.UpdatedOn.ToString();
            arr[7] = result.NotificationStatus;

            ListViewItem itm = new ListViewItem(arr);
            dlvCollection_Statement_return.Items.Add(itm);
        }

    }

    private void btnSubmitXML_Statement_Date_Click(object sender, EventArgs e)
    {
        BoomPartner.StatementsClient svc = new BoomPartner.StatementsClient("WCFStatements");
        BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
```

```
    PartnerInfo.Partner_username = txtXML_Statement_Date_PartnerUN.Text;
    PartnerInfo.Partner_password = txtXML_Statement_Date_PartnerPW.Text;

    DateTime StartDate;
    StartDate = new DateTime();
    StartDate = DateTime.ParseExact(txtXML_Statement_Date_Sdate.Text, "dd/MM/yy HH:mm:ss ", null);

    DateTime EndDate;
    EndDate = new DateTime();
    EndDate = DateTime.ParseExact(txtXML_Statement_Date_Edate.Text, "dd/MM/yy HH:mm:ss ", null);

    BoomPartner.searchcriteria SearchValues = new BoomPartner.searchcriteria();
    SearchValues.account_id = txtXML_Statement_Date_AccUsername.Text;
    SearchValues.message = txtXML_Statement_Date_Msg.Text;
    SearchValues.mobile_to = txtXML_Statement_Date_MobTo.Text;
    SearchValues.username = txtXML_Statement_Date_MsgUsername.Text;

    switch (cmbXML_Statement_Date_Dstatus.SelectedText.ToLower())
    {
        case "blank":
            SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.blank;
            break;
        case "failed":
            SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.failed;
            break;
        case "pending":
            SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.pending;
            break;
        case "success":
            SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.success;
            break;
        case "unknown":
            SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.unknown;
            break;
        default:
            SearchValues.status = Boom_SMS_API_Demo.BoomPartner.delivery_status.all;
            break;
    }

    txtXML_Statement_Date_return.Text = svc.XML_Statement_date(PartnerInfo, SearchValues, StartDate, EndDate);
}
```

```
        private void btnSubmit_Get_Subaccount_Credits_Dataset_Click(object sender, EventArgs e)
        {
            BoomPartner.StatementsClient svc = new BoomPartner.StatementsClient("WCFStatements");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtGet_Subaccount_Credits_Dataset_PartnerUN.Text;
            PartnerInfo.Partner_password = txtGet_Subaccount_Credits_Dataset_PartnerPW.Text;

            dgvGet_Subaccount_Credits_Dataset_return.DataSource = svc.Get_Subaccount_Credits_DataSet(PartnerInfo,
txtGet_Subaccount_Credits_Dataset_SmsAccountUn.Text);
        }

        private void btnSubmit_Get_Subaccount_Credits_Int_Click(object sender, EventArgs e)
        {
            BoomPartner.StatementsClient svc = new BoomPartner.StatementsClient("WCFStatements");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtGet_Subaccount_Credits_Int_PartnerUN.Text;
            PartnerInfo.Partner_password = txtGet_Subaccount_Credits_Int_PartnerPW.Text;

            txtGet_Subaccount_Credits_Int_return.Text = svc.Get_Subaccount_Credits_Int(PartnerInfo,
txtGet_Subaccount_Credits_Int_SmsAccountUn.Text).ToString();
        }

        private void btnSubmit_Get_Credit_Transactions_Dataset_Click(object sender, EventArgs e)
        {
            BoomPartner.StatementsClient svc = new BoomPartner.StatementsClient("WCFStatements");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtGet_Credit_Transactions_Dataset_PartnerUN.Text;
            PartnerInfo.Partner_password = txtGet_Credit_Transactions_Dataset_PartnerPW.Text;

            dgvGet_Credit_Transactions_Dataset_return.DataSource = svc.Get_Credit_Transactions_dataset(PartnerInfo);
        }

        private void btnSubmitGet_Credit_Transactions_XML_Click(object sender, EventArgs e)
        {
            BoomPartner.StatementsClient svc = new BoomPartner.StatementsClient("WCFStatements");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtGet_Credit_Transactions_XML_PartnerUN.Text;
            PartnerInfo.Partner_password = txtGet_Credit_Transactions_XML_PartnerPW.Text;

            txtGet_Credit_Transactions_XML_return.Text = svc.Get_Credit_Transactions_XML(PartnerInfo);
        }
```

```
        private void btnSubmit_Dataset_Failed_Messages_Date_Click(object sender, EventArgs e)
        {
            BoomPartner.StatementsClient svc = new BoomPartner.StatementsClient("WCFStatements");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtDataset_Failed_Messages_Date_PartnerUN.Text;
            PartnerInfo.Partner_password = txtDataset_Failed_Messages_Date_PartnerPW.Text;

            DateTime StartDate;
            StartDate = new DateTime();
            StartDate = DateTime.ParseExact(txtDataset_Failed_Messages_Date_Sdate.Text, "dd/MM/yy HH:mm:ss ", null);

            DateTime EndDate;
            EndDate = new DateTime();
            EndDate = DateTime.ParseExact(txtDataset_Failed_Messages_Date_Edate.Text, "dd/MM/yy HH:mm:ss ", null);

            BoomPartner.searchcriteria SearchValues = new BoomPartner.searchcriteria();
            SearchValues.account_id = txtDataset_Failed_Messages_Date_AccUsername.Text;
            SearchValues.message = txtDataset_Failed_Messages_Date_Msg.Text;
            SearchValues.mobile_to = txtDataset_Failed_Messages_Date_MobTo.Text;
            SearchValues.username = txtDataset_Failed_Messages_Date_MsgUsername.Text;

            dgvDataset_Failed_Messages_Date_return.DataSource = svc.Dataset_Failed_Messages_Date(PartnerInfo,
SearchValues, StartDate, EndDate);
        }

        private void btnSubmit_Dataset_Failed_Messages_Click(object sender, EventArgs e)
        {
            BoomPartner.StatementsClient svc = new BoomPartner.StatementsClient("WCFStatements");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtDataset_Failed_Messages_PartnerUN.Text;
            PartnerInfo.Partner_password = txtDataset_Failed_Messages_PartnerPW.Text;

            BoomPartner.searchcriteria SearchValues = new BoomPartner.searchcriteria();
            SearchValues.account_id = txtDataset_Failed_Messages_AccUsername.Text;
            SearchValues.message = txtDataset_Failed_Messages_Msg.Text;
            SearchValues.mobile_to = txtDataset_Failed_Messages_MobTo.Text;
            SearchValues.username = txtDataset_Failed_Messages_MsgUsername.Text;

            dgvDataset_Failed_Messages_return.DataSource = svc.Dataset_Failed_Messages(PartnerInfo, SearchValues);
        }

        private void btnSubmit_XML_Failed_Messages_Date_Click(object sender, EventArgs e)
```

```
        {
            BoomPartner.StatementsClient svc = new BoomPartner.StatementsClient("WCFStatements");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtXML_Failed_Messages_Date_PartnerUN.Text;
            PartnerInfo.Partner_password = txtXML_Failed_Messages_Date_PartnerPW.Text;

            DateTime StartDate;
            StartDate = new DateTime();
            StartDate = DateTime.ParseExact(txtXML_Failed_Messages_Date_Sdate.Text, "dd/MM/yy HH:mm:ss ", null);

            DateTime EndDate;
            EndDate = new DateTime();
            EndDate = DateTime.ParseExact(txtXML_Failed_Messages_Date_Edate.Text, "dd/MM/yy HH:mm:ss ", null);

            BoomPartner.searchcriteria SearchValues = new BoomPartner.searchcriteria();
            SearchValues.account_id = txtXML_Failed_Messages_Date_AccUsername.Text;
            SearchValues.message = txtXML_Failed_Messages_Date_Msg.Text;
            SearchValues.mobile_to = txtXML_Failed_Messages_Date_MobTo.Text;
            SearchValues.username = txtXML_Failed_Messages_Date_MsgUsername.Text;

            txtXML_Failed_Messages_Date_return.Text = svc.XML_Failed_Messages_Date(PartnerInfo, SearchValues, StartDate,
EndDate);
        }

        private void btnSubmit_XML_Failed_Messages_Click(object sender, EventArgs e)
        {
            BoomPartner.StatementsClient svc = new BoomPartner.StatementsClient("WCFStatements");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtXML_Failed_Messages_PartnerUN.Text;
            PartnerInfo.Partner_password = txtXML_Failed_Messages_PartnerPW.Text;

            BoomPartner.searchcriteria SearchValues = new BoomPartner.searchcriteria();
            SearchValues.account_id = txtXML_Failed_Messages_AccUsername.Text;
            SearchValues.message = txtXML_Failed_Messages_Msg.Text;
            SearchValues.mobile_to = txtXML_Failed_Messages_MobTo.Text;
            SearchValues.username = txtXML_Failed_Messages_MsgUsername.Text;

            txtXML_Failed_Messages_return.Text = svc.XML_Failed_Messages(PartnerInfo, SearchValues);
        }

        private void btnSubmit_txtAccount_Create_Click(object sender, EventArgs e)
        {
```

```
            BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtAccount_Create_PartnerUN.Text;
            PartnerInfo.Partner_password = txtAccount_Create_PartnerPW.Text;


            switch (svc.Account_Create(PartnerInfo, txtAccount_Create_Username.Text, txtAccount_Create_Password.Text,
txtAccount_Create_Email.Text, txtAccount_Create_ContactFirstName.Text, txtAccount_Create_ContactLastName.Text,
txtAccount_Create_ContactTel.Text, txtAccount_Create_SMSSenderId.Text, txtAccount_Update_CompanyName.Text,
txtAccount_Create_Email.Text, txtAccount_Create_NotifyURL.Text))
            {
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.EmptyValue:
                    txtAccount_Create_return.Text = "EmptyValue";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.FieldNotUnique:
                    txtAccount_Create_return.Text = "FieldNotUnique";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InternalError:
                    txtAccount_Create_return.Text = "InternalError";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidAuthentication:
                    txtAccount_Create_return.Text = "InvalidAuthentication";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidCharacters:
                    txtAccount_Create_return.Text = "InvalidCharacters";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidDateField:
                    txtAccount_Create_return.Text = "InvalidDateField";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidEmail:
                    txtAccount_Create_return.Text = "InvalidEmail";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidField:
                    txtAccount_Create_return.Text = "InvalidField";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidPartner:
                    txtAccount_Create_return.Text = "InvalidPartner";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.PasswordMismatch:
                    txtAccount_Create_return.Text = "PasswordMismatch";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.Success:
```

```
                    txtAccount_Create_return.Text = "Success";
                    break;
                default:
                    txtAccount_Create_return.Text = "Unexpected Result";
                    break;
            }
        }

        private void btnSubmit_txtTransfer_Credits_Click(object sender, EventArgs e)
        {
            BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtTransfer_Credits_PartnerUN.Text;
            PartnerInfo.Partner_password = txtTransfer_Credits_PartnerPW.Text;

            if
(svc.Transfer_Credits(PartnerInfo,txtTransfer_Credits_SmsAccountUN.Text,Convert.ToInt32(txtTransfer_Credits_Amount.Text))
)
            {
                txtTransfer_Credits_return.Text = "True";
            }
                else
            {
                txtTransfer_Credits_return.Text = "False";
            }

        }

        private void btnSubmit_txtAccount_Change_Password_Click(object sender, EventArgs e)
        {
            BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtAccount_Change_Password_PartnerUN.Text;
            PartnerInfo.Partner_password = txtAccount_Change_Password_PartnerPW.Text;

            switch
(svc.Account_Change_Password(PartnerInfo,txtAccount_Change_Password_SmsAccountUN.Text,txtAccount_Change_Password_OldPass.
Text,txtAccount_Change_Password_NewPass.Text))
            {
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.EmptyValue:
                    txtAccount_Change_Password_return.Text = "EmptyValue";
                    break;
```

```
            case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.FieldNotUnique:
                txtAccount_Change_Password_return.Text = "FieldNotUnique";
                break;
            case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InternalError:
                txtAccount_Change_Password_return.Text = "InternalError";
                break;
            case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidAuthentication:
                txtAccount_Change_Password_return.Text = "InvalidAuthentication";
                break;
            case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidCharacters:
                txtAccount_Change_Password_return.Text = "InvalidCharacters";
                break;
            case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidDateField:
                txtAccount_Change_Password_return.Text = "InvalidDateField";
                break;
            case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidEmail:
                txtAccount_Change_Password_return.Text = "InvalidEmail";
                break;
            case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidField:
                txtAccount_Change_Password_return.Text = "InvalidField";
                break;
            case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidPartner:
                txtAccount_Change_Password_return.Text = "InvalidPartner";
                break;
            case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.PasswordMismatch:
                txtAccount_Change_Password_return.Text = "PasswordMismatch";
                break;
            case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.Success:
                txtAccount_Change_Password_return.Text = "Success";
                break;
            default:
                txtAccount_Change_Password_return.Text = "Unexpected Result";
                break;
        }

    }

    private void btnSubmit_txtAccount_Update_Click(object sender, EventArgs e)
    {
        BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
        BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
        PartnerInfo.Partner_username = txtAccount_Update_PartnerUN.Text;
```

```
            PartnerInfo.Partner_password = txtAccount_Update_PartnerPW.Text;


            switch (svc.Account_Update(PartnerInfo, txtAccount_Update_Username.Text, txtAccount_Update_Email.Text,
txtAccount_Update_ContactFirstName.Text, txtAccount_Update_ContactLastName.Text, txtAccount_Update_ContactTel.Text,
txtAccount_Update_SmsSenderId.Text, txtAccount_Update_CompanyName.Text, txtAccount_Update_Email.Text,
txtAccount_Update_NotifyURL.Text,chkAdd_Subaccount_Contact_NotifySent.Checked,chkAdd_Subaccount_Contact_NotifyFail.Checke
d,chkAdd_Subaccount_Contact_NotifySent.Checked,chkAdd_Subaccount_Contact_NotifyInvoice.Checked,chkAdd_Subaccount_Contact_
Notifyworldpay.Checked,chkAdd_Subaccount_Contact_NotifyWeekly.Checked,chkAdd_Subaccount_Contact_NotifyLowCredits.Checked)
)
            {
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.EmptyValue:
                    txtAccount_Update_return.Text = "EmptyValue";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.FieldNotUnique:
                    txtAccount_Update_return.Text = "FieldNotUnique";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InternalError:
                    txtAccount_Update_return.Text = "InternalError";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidAuthentication:
                    txtAccount_Update_return.Text = "InvalidAuthentication";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidCharacters:
                    txtAccount_Update_return.Text = "InvalidCharacters";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidDateField:
                    txtAccount_Update_return.Text = "InvalidDateField";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidEmail:
                    txtAccount_Update_return.Text = "InvalidEmail";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidField:
                    txtAccount_Update_return.Text = "InvalidField";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidPartner:
                    txtAccount_Update_return.Text = "InvalidPartner";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.PasswordMismatch:
                    txtAccount_Update_return.Text = "PasswordMismatch";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.Success:
```

```
                    txtAccount_Update_return.Text = "Success";
                    break;
                default:
                    txtAccount_Update_return.Text = "Unexpected Result";
                    break;
            }
        }

        private void btnSubmit_txtGet_Subaccount_Details_Dataset_Click(object sender, EventArgs e)
        {
            BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtGet_Subaccount_Details_Dataset_PartnerUN.Text;
            PartnerInfo.Partner_password = txtGet_Subaccount_Details_Dataset_PartnerPW.Text;

            dgvGet_Subaccount_Details_Dataset_return.DataSource = svc.Get_Subaccount_Details_Dataset(PartnerInfo,
txtGet_Subaccount_Details_Dataset_Username.Text);
        }

        private void btnSubmit_Get_Subaccount_Details_XML_Click(object sender, EventArgs e)
        {
            BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtGet_Subaccount_Details_XML_PartnerUN.Text;
            PartnerInfo.Partner_password = txtGet_Subaccount_Details_XML_PartnerPW.Text;

            txtGet_Subaccount_Details_XML_return.Text = svc.Get_Subaccount_Details_XML(PartnerInfo,
txtGet_Subaccount_Details_XML_Username.Text);
        }

        private void btnSubmit_txtGet_Subaccount_Contacts_Dataset_Click(object sender, EventArgs e)
        {
            BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtGet_Subaccount_Contacts_Dataset_PartnerUN.Text;
            PartnerInfo.Partner_password = txtGet_Subaccount_Contacts_Dataset_PartnerPW.Text;

            dgvGet_Subaccount_Contacts_Dataset_return.DataSource = svc.Get_Subaccount_Contacts_Dataset(PartnerInfo,
txtGet_Subaccount_Contacts_Dataset_ContactName.Text);
        }

        private void btnSubmit_txtGet_Subaccount_Contacts_XML_Click(object sender, EventArgs e)
```

```
        {
            BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtGet_Subaccount_Contacts_XML_PartnerUN.Text;
            PartnerInfo.Partner_password = txtGet_Subaccount_Contacts_XML_PartnerPW.Text;

            txtGet_Subaccount_Contacts_XML_return.Text = svc.Get_Subaccount_Contacts_XML(PartnerInfo,
txtGet_Subaccount_Contacts_XML_ContactName.Text);
        }

        private void btnSubmit_txtRemove_Subaccount_Contact_Click(object sender, EventArgs e)
        {
            BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtRemove_Subaccount_Contact_PartnerUN.Text;
            PartnerInfo.Partner_password = txtRemove_Subaccount_Contact_PartnerPW.Text;

            switch
(svc.Remove_Subaccount_Contact(PartnerInfo,txtRemove_Subaccount_Contact_Username.Text,txtRemove_Subaccount_Contact_Contac
tName.Text))
            {
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.EmptyValue:
                    txtRemove_Subaccount_Contact_return.Text = "EmptyValue";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.FieldNotUnique:
                    txtRemove_Subaccount_Contact_return.Text = "FieldNotUnique";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InternalError:
                    txtRemove_Subaccount_Contact_return.Text = "InternalError";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidAuthentication:
                    txtRemove_Subaccount_Contact_return.Text = "InvalidAuthentication";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidCharacters:
                    txtRemove_Subaccount_Contact_return.Text = "InvalidCharacters";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidDateField:
                    txtRemove_Subaccount_Contact_return.Text = "InvalidDateField";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidEmail:
                    txtRemove_Subaccount_Contact_return.Text = "InvalidEmail";
                    break;
```

```
            case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidField:
                txtRemove_Subaccount_Contact_return.Text = "InvalidField";
                break;
            case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidPartner:
                txtRemove_Subaccount_Contact_return.Text = "InvalidPartner";
                break;
            case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.PasswordMismatch:
                txtRemove_Subaccount_Contact_return.Text = "PasswordMismatch";
                break;
            case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.Success:
                txtRemove_Subaccount_Contact_return.Text = "Success";
                break;
            default:
                txtRemove_Subaccount_Contact_return.Text = "Unexpected Result";
                break;
        }

    }

    private void btnSubmit_Add_Subaccount_Contact_Click(object sender, EventArgs e)
    {
        BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
        BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
        PartnerInfo.Partner_username = txtAdd_Subaccount_Contact_PartnerUN.Text;
        PartnerInfo.Partner_password = txtAdd_Subaccount_Contact_PartnerPW.Text;

        BoomPartner.ContactType ContactType;

        switch (cmbAdd_Subaccount_Contact_Type.SelectedText.ToLower())
        {
            case "accounts":
                ContactType = Boom_SMS_API_Demo.BoomPartner.ContactType.Accounts;
                break;
            case "partner":
                ContactType = Boom_SMS_API_Demo.BoomPartner.ContactType.Partner;
                break;
            case "techical":
                ContactType = Boom_SMS_API_Demo.BoomPartner.ContactType.Technical;
                break;
            default:
                ContactType = Boom_SMS_API_Demo.BoomPartner.ContactType.Other;
                break;
```

```
            }

            switch (svc.Add_Subaccount_Contact(PartnerInfo, txtAdd_Subaccount_Contact_Username.Text,
txtAdd_Subaccount_Contact_FirstName.Text, txtAdd_Subaccount_Contact_Email.Text, txtAdd_Subaccount_Contact_Tel.Text,
ContactType, txtAdd_Subaccount_Contact_LastName.Text, txtAdd_Subaccount_Contact_Mob.Text,
txtAdd_Subaccount_Contact_Address.Text, txtAdd_Subaccount_Contact_Postcode.Text,
chkAdd_Subaccount_Contact_NotifySent.Checked, chkAdd_Subaccount_Contact_NotifyFail.Checked,
chkAdd_Subaccount_Contact_NotifySent.Checked, chkAdd_Subaccount_Contact_NotifyInvoice.Checked,
chkAdd_Subaccount_Contact_Notifyworldpay.Checked, chkAdd_Subaccount_Contact_NotifyWeekly.Checked,
chkAdd_Subaccount_Contact_NotifyLowCredits.Checked))
            {
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.EmptyValue:
                    txtAdd_Subaccount_Contact_return.Text = "EmptyValue";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.FieldNotUnique:
                    txtAdd_Subaccount_Contact_return.Text = "FieldNotUnique";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InternalError:
                    txtAdd_Subaccount_Contact_return.Text = "InternalError";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidAuthentication:
                    txtAdd_Subaccount_Contact_return.Text = "InvalidAuthentication";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidCharacters:
                    txtAdd_Subaccount_Contact_return.Text = "InvalidCharacters";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidDateField:
                    txtAdd_Subaccount_Contact_return.Text = "InvalidDateField";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidEmail:
                    txtAdd_Subaccount_Contact_return.Text = "InvalidEmail";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidField:
                    txtAdd_Subaccount_Contact_return.Text = "InvalidField";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.InvalidPartner:
                    txtAdd_Subaccount_Contact_return.Text = "InvalidPartner";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.PasswordMismatch:
                    txtAdd_Subaccount_Contact_return.Text = "PasswordMismatch";
                    break;
                case Boom_SMS_API_Demo.BoomPartner.Account_Operation_Result.Success:
```

```
                txtAdd_Subaccount_Contact_return.Text = "Success";
                break;
            default:
                txtAdd_Subaccount_Contact_return.Text = "Unexpected Result";
                break;
        }
    }

    private void btnSubmit_Get_Subaccounts_Dataset_Click(object sender, EventArgs e)
    {
        BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
        BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
        PartnerInfo.Partner_username = txtGet_Subaccounts_Dataset_PartnerUN.Text;
        PartnerInfo.Partner_password = txtGet_Subaccounts_Dataset_PartnerPW.Text;

        dgvGet_Subaccounts_Dataset_return.DataSource = svc.Get_Subaccounts_DataSet(PartnerInfo);
    }

    private void btnSubmit_Get_Subaccounts_XML_Click(object sender, EventArgs e)
    {
        BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
        BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
        PartnerInfo.Partner_username = txtGet_Subaccounts_XML_PartnerUN.Text;
        PartnerInfo.Partner_password = txtGet_Subaccounts_XML_PartnerPW.Text;

        txtGet_Subaccounts_XML_return.Text = svc.Get_Subaccounts_XML(PartnerInfo);
    }

    private void btnSubmit_Get_Subaccounts_Latest_Messages_Dataset_Click(object sender, EventArgs e)
    {
        BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
        BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
        PartnerInfo.Partner_username = txtGet_Subaccounts_Latest_Messages_Dataset_PartnerUN.Text;
        PartnerInfo.Partner_password = txtGet_Subaccounts_Latest_Messages_Dataset_PartnerPW.Text;

        dgvGet_Subaccounts_Latest_Messages_Dataset_return.DataSource =
svc.Get_Subaccounts_Latest_Messages_DataSet(PartnerInfo);
    }

    private void btnSubmit_Get_Subaccounts_Latest_Messages_XML_Click(object sender, EventArgs e)
    {
        BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
```

```
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtGet_Subaccounts_Latest_Messages_XML_PartnerUN.Text;
            PartnerInfo.Partner_password = txtGet_Subaccounts_Latest_Messages_XML_PartnerPW.Text;

            txtGet_Subaccounts_Latest_Messages_XML_return.Text = svc.Get_Subaccounts_Latest_Messages_XML(PartnerInfo);
        }

        private void btnSubmit_Get_Inactive_Subaccounts_Dataset_Click(object sender, EventArgs e)
        {
            BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtGet_Inactive_Subaccounts_Dataset_PartnerUN.Text;
            PartnerInfo.Partner_password = txtGet_Inactive_Subaccounts_Dataset_PartnerPW.Text;

            BoomPartner.timeframe Timespan;

            switch (cmbGet_Inactive_Subaccounts_Dataset_Timeframe.SelectedText.ToLower())
            {
                case "day":
                    Timespan = Boom_SMS_API_Demo.BoomPartner.timeframe.Day;
                    break;
                case "month":
                    Timespan = Boom_SMS_API_Demo.BoomPartner.timeframe.Month;
                    break;
                case "week":
                    Timespan = Boom_SMS_API_Demo.BoomPartner.timeframe.Week;
                    break;
                default:
                    Timespan = Boom_SMS_API_Demo.BoomPartner.timeframe.Year;
                    break;
            }

            dgvGet_Inactive_Subaccounts_Dataset_return.DataSource = svc.Get_Inactive_Subaccounts_DataSet(PartnerInfo,
Timespan);
        }

        private void btnSubmit_Get_Inactive_Subaccounts_XML_Click(object sender, EventArgs e)
        {
            BoomPartner.Account_ManagementClient svc = new BoomPartner.Account_ManagementClient("WCFAccount_Managment");
            BoomPartner.Partner_credentials PartnerInfo = new BoomPartner.Partner_credentials();
            PartnerInfo.Partner_username = txtGet_Inactive_Subaccounts_XML_PartnerUN.Text;
            PartnerInfo.Partner_password = txtGet_Inactive_Subaccounts_XML_PartnerPW.Text;
```

```
    BoomPartner.timeframe Timespan;

    switch (cmbGet_Inactive_Subaccounts_Dataset_Timeframe.SelectedText.ToLower())
    {
        case "day":
            Timespan = Boom_SMS_API_Demo.BoomPartner.timeframe.Day;
            break;
        case "month":
            Timespan = Boom_SMS_API_Demo.BoomPartner.timeframe.Month;
            break;
        case "week":
            Timespan = Boom_SMS_API_Demo.BoomPartner.timeframe.Week;
            break;
        default:
            Timespan = Boom_SMS_API_Demo.BoomPartner.timeframe.Year;
            break;
    }

    txtGet_Inactive_Subaccounts_XML_return.Text = svc.Get_Inactive_Subaccounts_XML(PartnerInfo, Timespan);
}

private void btnSubmit_XML_String_Send_Click(object sender, EventArgs e)
{
    BoomUser.SendingClient svc = new BoomUser.SendingClient("SendingWCF");

    if (svc.XML_String_Send(txtPost_String_XML_String_Send.Text))
    {
        txtXML_String_Send_Return.Text = "True";
    }
    else
    {
        txtXML_String_Send_Return.Text = "False";
    }


}

private void btnSubmit_SMS_Send_Object_Click(object sender, EventArgs e)
{
    BoomUser.SendingClient svc = new BoomUser.SendingClient("SendingHTTP");
    BoomUser.smsclass[] MessageArray = new BoomUser.smsclass[1];
```

```
    string[] Numbers = new string[1];

    MessageArray[0] = new BoomUser.smsclass();
    MessageArray[0].Account_ID = txtSMS_Send_Object_AccountID.Text;
    MessageArray[0].Password = txtSMS_Send_Object_Password.Text;
    MessageArray[0].Username = txtSMS_Send_Object_Username.Text;
    Numbers[0] = txtSMS_Send_Object_MobTo.Text;
    MessageArray[0].Mobile_To = Numbers;
    MessageArray[0].Message = txtSMS_Send_Object_Message.Text;
    MessageArray[0].Notify = chkSMS_Send_Object_Notify.Checked;
    MessageArray[0].Sender_Email = txtSMS_Send_Object_SenderEmail.Text;

    if (svc.SMS_Send_Object_Test(MessageArray))
    {
        txtSMS_Send_Object_Return.Text = "True";
    }
    else
    {
        txtSMS_Send_Object_Return.Text = "False";
    }
}

private void btnSubmit_XML_String_Send_Delayed_Click(object sender, EventArgs e)
{
    BoomUser.SendingClient svc = new BoomUser.SendingClient("SendingWCF");

    DateTime MyDateTime;
    MyDateTime = new DateTime();
    MyDateTime = DateTime.ParseExact(txtTimeToSend_XML_String_Send_Delayed.Text, "dd/MM/yy HH:mm:ss ",null);

    if (svc.XML_String_Send_Delayed(txtPost_String_XML_String_Send_Delayed.Text, MyDateTime))
    {
        txtXML_String_Send_Delayed_Return.Text = "True";
    }
    else
    {
        txtXML_String_Send_Delayed_Return.Text = "False";
    }
}

private void btnSubmit_SMS_Send_Object_Delayed_Click(object sender, EventArgs e)
{
```

```
    BoomUser.SendingClient svc = new BoomUser.SendingClient("SendingWCF");
    BoomUser.smsclass[] MessageArray = new BoomUser.smsclass[1];
    string[] Numbers = new string[1];

    DateTime MyDateTime;
    MyDateTime = new DateTime();
    MyDateTime = DateTime.ParseExact(txtTimeToSend_SMS_Send_Object_Delayed.Text, "dd/MM/yy HH:mm:ss ",null);

    MessageArray[0] = new BoomUser.smsclass();
    MessageArray[0].Account_ID = txtSMS_Send_Object_Delayed_AccountID.Text;
    MessageArray[0].Password = txtSMS_Send_Object_Delayed_Password.Text;
    MessageArray[0].Username = txtSMS_Send_Object_Delayed_Username.Text;
    Numbers[0] = txtSMS_Send_Object_Delayed_MobTo.Text;
    MessageArray[0].Mobile_To = Numbers;
    MessageArray[0].Message = txtSMS_Send_Object_Delayed_Message.Text;
    MessageArray[0].Notify = chkSMS_Send_Object_Delayed_Notify.Checked;
    MessageArray[0].Sender_Email = txtSMS_Send_Object_Delayed_SenderEmail.Text;

    if (svc.SMS_Send_Object_Delayed(MessageArray, MyDateTime))
    {
        txtSMS_Send_Object_Delayed_Return.Text = "True";
    }
    else
    {
        txtSMS_Send_Object_Delayed_Return.Text = "False";
    }
}

private void btnSubmit_Credential_Check_Click(object sender, EventArgs e)
{
    BoomUser.CheckingClient svc = new BoomUser.CheckingClient("CheckingWCF");

    switch (svc.Credential_Check(txtCredential_Check_UsrName.Text, txtCredential_Check_Passwd.Text))
    {
        case BoomUser.CredCheck.noMatch:
            txtCredential_Check_return.Text = "No Match";
            break;
        case BoomUser.CredCheck.partMatch:
            txtCredential_Check_return.Text = "Part Match";
            break;
        case BoomUser.CredCheck.fullMatch:
            txtCredential_Check_return.Text = "Full Match";
```

```
                    break;
                default:
                    txtCredential_Check_return.Text = "Unexpected Response";
                    break;
            }
        }

        private void btnSubmit_Credit_Check_Click(object sender, EventArgs e)
        {
            BoomUser.CheckingClient svc = new BoomUser.CheckingClient("CheckingWCF");

            txtCredit_Check_return.Text = svc.Credit_Check(txtCredit_Check_UsrName.Text,
txtCredit_Check_Passwd.Text).ToString();
        }

        private void btnSubmit_Get_Invoices_Dataset_Click(object sender, EventArgs e)
        {
            BoomUser.CheckingClient svc = new BoomUser.CheckingClient("CheckingWCF");

            dgvGet_Invoices_Dataset_return.DataSource = svc.Get_Invoices_Dataset(txtGet_Invoices_Dataset_UsrName.Text,
txtGet_Invoices_Dataset_Passwd.Text).Tables[0];
        }

        private void button2_Click(object sender, EventArgs e)
        {
            BoomUser.CheckingClient svc = new BoomUser.CheckingClient("CheckingWCF");

            txtGet_Invoices_XML_return.Text = svc.Get_Invoices_XML(txtGet_Invoices_XML_UsrName.Text,
txtGet_Invoices_XML_Passwd.Text);
        }

        private void btnSubmit_Get_Unpaid_Invoices_Dataset_Click(object sender, EventArgs e)
        {
            BoomUser.CheckingClient svc = new BoomUser.CheckingClient("CheckingWCF");

            dgvGet_Unpaid_Invoices_Dataset_return.DataSource =
svc.Get_Unpaid_Invoices_Dataset(txtGet_Unpaid_Invoices_Dataset_UsrName.Text,
txtGet_Unpaid_Invoices_Dataset_Passwd.Text).Tables[0];
        }

        private void btnSubmit_Get_Unpaid_Invoices_XML_Click(object sender, EventArgs e)
        {
```

```
            BoomUser.CheckingClient svc = new BoomUser.CheckingClient("CheckingWCF");

            txtGet_Unpaid_Invoices_XML_return.Text = svc.Get_Unpaid_Invoices_XML(txtGet_Unpaid_Invoices_XML_UsrName.Text,
txtGet_Unpaid_Invoices_XML_Passwd.Text);
        }

        private void btnSubmit_Get_Account_Details_Dataset_Click(object sender, EventArgs e)
        {
            BoomUser.DetailsClient svc = new BoomUser.DetailsClient("DetailsWCF");

            dgvGet_Account_Details_Dataset_return.DataSource =
svc.Get_Account_Details_Dataset(txtGet_Account_Details_Dataset_UsrName.Text,
txtGet_Account_Details_Dataset_Passwd.Text).Tables[0];
        }

        private void btnSubmit_Get_Account_Details_XML_Click(object sender, EventArgs e)
        {
            BoomUser.DetailsClient svc = new BoomUser.DetailsClient("DetailsWCF");

            txtGet_Account_Details_XML_return.Text = svc.Get_Account_Details_XML(txtGet_Account_Details_XML_UsrName.Text,
txtGet_Account_Details_XML_Passwd.Text);
        }

        private void btnSubmit_Get_Messages_Dataset_Click(object sender, EventArgs e)
        {
            BoomUser.DetailsClient svc = new BoomUser.DetailsClient("DetailsWCF");

            DateTime StartDate;
            StartDate = new DateTime();
            StartDate = DateTime.ParseExact(txtGet_Messages_Dataset_Sdate.Text, "dd/MM/yy HH:mm:ss ", null);
            DateTime EndDate;
            EndDate = new DateTime();
            EndDate = DateTime.ParseExact(txtGet_Messages_Dataset_Edate.Text, "dd/MM/yy HH:mm:ss ", null);

            BoomUser.searchcriteria SearchValues;
            SearchValues = new BoomUser.searchcriteria();

            SearchValues.mobile_to = txtGet_Messages_Dataset_mobto.Text;
            SearchValues.message = txtGet_Messages_Dataset_msg.Text;

            switch (cmbGet_Messages_Dataset_dstatus.SelectedText.ToLower())
            {
```

```
                case "blank":
                    SearchValues.status = Boom_SMS_API_Demo.BoomUser.delivery_status.blank;
                    break;
                case "failed":
                    SearchValues.status = Boom_SMS_API_Demo.BoomUser.delivery_status.failed;
                    break;
                case "pending":
                    SearchValues.status = Boom_SMS_API_Demo.BoomUser.delivery_status.pending;
                    break;
                case "success":
                    SearchValues.status = Boom_SMS_API_Demo.BoomUser.delivery_status.success;
                    break;
                case "unknown":
                    SearchValues.status = Boom_SMS_API_Demo.BoomUser.delivery_status.unknown;
                    break;
                default:
                    SearchValues.status = Boom_SMS_API_Demo.BoomUser.delivery_status.all;
                    break;
            }

            dgvGet_Messages_Dataset_return.DataSource = svc.Get_Messages_Dataset(txtGet_Messages_Dataset_UsrName.Text,
txtGet_Messages_Dataset_Passwd.Text,SearchValues,StartDate,EndDate).Tables[0];
        }

        private void btnSubmit_Get_Messages_XML_Click(object sender, EventArgs e)
        {
            BoomUser.DetailsClient svc = new BoomUser.DetailsClient("DetailsWCF");

            DateTime StartDate;
            StartDate = new DateTime();
            StartDate = DateTime.ParseExact(txtGet_Messages_XML_Sdate.Text, "dd/MM/yy HH:mm:ss ", null);
            DateTime EndDate;
            EndDate = new DateTime();
            EndDate = DateTime.ParseExact(txtGet_Messages_XML_Edate.Text, "dd/MM/yy HH:mm:ss ", null);

            BoomUser.searchcriteria SearchValues;
            SearchValues = new BoomUser.searchcriteria();

            SearchValues.mobile_to = txtGet_Messages_XML_mobto.Text;
            SearchValues.message = txtGet_Messages_XML_msg.Text;

            switch (cmbGet_Messages_XML_dstatus.SelectedText.ToLower())
```

```
            {
                case "blank":
                    SearchValues.status = Boom_SMS_API_Demo.BoomUser.delivery_status.blank;
                    break;
                case "failed":
                    SearchValues.status = Boom_SMS_API_Demo.BoomUser.delivery_status.failed;
                    break;
                case "pending":
                    SearchValues.status = Boom_SMS_API_Demo.BoomUser.delivery_status.pending;
                    break;
                case "success":
                    SearchValues.status = Boom_SMS_API_Demo.BoomUser.delivery_status.success;
                    break;
                case "unknown":
                    SearchValues.status = Boom_SMS_API_Demo.BoomUser.delivery_status.unknown;
                    break;
                default:
                    SearchValues.status = Boom_SMS_API_Demo.BoomUser.delivery_status.all;
                    break;
            }

            txtGet_Messages_XML_return.Text = svc.Get_Messages_XML(txtGet_Messages_XML_UsrName.Text,
txtGet_Messages_XML_Passwd.Text, SearchValues, StartDate, EndDate);
        }

        private void btnSubmit_Get_Contacts_Dataset_Click(object sender, EventArgs e)
        {
            BoomUser.DetailsClient svc = new BoomUser.DetailsClient("DetailsWCF");

            dgvGet_Contacts_Dataset_return.DataSource = svc.Get_Contacts_Dataset(txtGet_Contacts_Dataset_UsrName.Text,
txtGet_Contacts_Dataset_Passwd.Text).Tables[0];
        }

        private void btnSubmit_Get_Contacts_XML_Click(object sender, EventArgs e)
        {
            BoomUser.DetailsClient svc = new BoomUser.DetailsClient("DetailsWCF");

            txtGet_Contacts_XML_return.Text = svc.Get_Contacts_XML(txtGet_Contacts_XML_UsrName.Text,
txtGet_Contacts_XML_Passwd.Text);
        }

        private void btnSubmit_Add_Contact_Click(object sender, EventArgs e)
```

```
        {
            BoomUser.DetailsClient svc = new BoomUser.DetailsClient("DetailsWCF");

            BoomUser.ContactType TempContactType;
            TempContactType = new BoomUser.ContactType();

            switch (cmbAdd_Contact_Type.SelectedText.ToLower())
            {
                case "accounts":
                    TempContactType = Boom_SMS_API_Demo.BoomUser.ContactType.Accounts;
                    break;
                case "technical":
                    TempContactType = Boom_SMS_API_Demo.BoomUser.ContactType.Technical;
                    break;
                case "partner":
                    TempContactType = Boom_SMS_API_Demo.BoomUser.ContactType.Partner;
                    break;
                default:
                    TempContactType = Boom_SMS_API_Demo.BoomUser.ContactType.Other;
                    break;
            }

            switch
(svc.Add_Contact(txtAdd_Contact_UsrName.Text,txtCredential_Check_Passwd.Text,txtAdd_Contact_FirstName.Text,txtAdd_Contact
_Email.Text,txtAdd_Contact_Tel.Text,TempContactType,txtAdd_Contact_LastName.Text,txtAdd_Contact_Mob.Text,txtAdd_Contact_A
ddress.Text,txtAdd_Contact_Postcode.Text,chkAdd_Contact_NotifySent.Checked,chkAdd_Contact_NotifyFail.Checked,chkAdd_Conta
ct_NotifyReceived.Checked,chkAdd_Contact_NotifyInvoice.Checked,chkAdd_Contact_NotifyWP.Checked,chkAdd_Contact_NotifyWeekl
y.Checked,chkAdd_Contact_NotifyLowcreds.Checked))
            {
                case BoomUser.Account_Operation_Result.EmptyValue:
                    txtAdd_Contact_return.Text = "Empty Value";
                    break;
                case BoomUser.Account_Operation_Result.FieldNotUnique:
                    txtAdd_Contact_return.Text = "Field not Unique";
                    break;
                case BoomUser.Account_Operation_Result.InternalError:
                    txtAdd_Contact_return.Text = "Internal Error";
                    break;
                case BoomUser.Account_Operation_Result.InvalidAuthentication:
                    txtAdd_Contact_return.Text = "Invalid Authentication";
                    break;
                case BoomUser.Account_Operation_Result.InvalidCharacters:
```

```
                    txtAdd_Contact_return.Text = "Invalid Characters";
                    break;
                case BoomUser.Account_Operation_Result.InvalidDateField:
                    txtAdd_Contact_return.Text = "Invalid Date Field";
                    break;
                case BoomUser.Account_Operation_Result.InvalidEmail:
                    txtAdd_Contact_return.Text = "Invalid Email";
                    break;
                case BoomUser.Account_Operation_Result.InvalidField:
                    txtAdd_Contact_return.Text = "Invalid field";
                    break;
                case BoomUser.Account_Operation_Result.InvalidPartner:
                    txtAdd_Contact_return.Text = "Invalid Partner";
                    break;
                case BoomUser.Account_Operation_Result.PasswordMismatch:
                    txtAdd_Contact_return.Text = "Password Mismatch";
                    break;
                case BoomUser.Account_Operation_Result.Success:
                    txtAdd_Contact_return.Text = "Success";
                    break;
                default:
                    txtAdd_Contact_return.Text = "Unexpected Result";
                    break;
            }

        }

        private void btnSubmit_Remove_Contact_Click(object sender, EventArgs e)
        {
            BoomUser.DetailsClient svc = new BoomUser.DetailsClient("DetailsWCF");

            switch
(svc.Remove_Contact(txtRemove_Contact_UsrName.Text,txtRemove_Contact_Passwd.Text,txtRemove_Contact_ContactName.Text))
            {
                case BoomUser.Account_Operation_Result.EmptyValue:
                    txtRemove_Contact_return.Text = "Empty Value";
                    break;
                case BoomUser.Account_Operation_Result.FieldNotUnique:
                    txtRemove_Contact_return.Text = "Field not Unique";
                    break;
                case BoomUser.Account_Operation_Result.InternalError:
                    txtRemove_Contact_return.Text = "Internal Error";
```

```
                break;
            case BoomUser.Account_Operation_Result.InvalidAuthentication:
                txtRemove_Contact_return.Text = "Invalid Authentication";
                break;
            case BoomUser.Account_Operation_Result.InvalidCharacters:
                txtRemove_Contact_return.Text = "Invalid Characters";
                break;
            case BoomUser.Account_Operation_Result.InvalidDateField:
                txtRemove_Contact_return.Text = "Invalid Date Field";
                break;
            case BoomUser.Account_Operation_Result.InvalidEmail:
                txtRemove_Contact_return.Text = "Invalid Email";
                break;
            case BoomUser.Account_Operation_Result.InvalidField:
                txtRemove_Contact_return.Text = "Invalid field";
                break;
            case BoomUser.Account_Operation_Result.InvalidPartner:
                txtRemove_Contact_return.Text = "Invalid Partner";
                break;
            case BoomUser.Account_Operation_Result.PasswordMismatch:
                txtRemove_Contact_return.Text = "Password Mismatch";
                break;
            case BoomUser.Account_Operation_Result.Success:
                txtRemove_Contact_return.Text = "Success";
                break;
            default:
                txtRemove_Contact_return.Text = "Unexpected Result";
                break;
        }
    }

    private void btnSubmit_Get_Templates_Dataset_Click(object sender, EventArgs e)
    {
        BoomUser.DetailsClient svc = new BoomUser.DetailsClient("DetailsWCF");

        dgvGet_Templates_Dataset_return.DataSource = svc.Get_Templates_Dataset(txtGet_Templates_Dataset_UsrName.Text,
txtGet_Templates_Dataset_Passwd.Text).Tables[0];
    }

    private void btnSubmit_Get_Templates_XML_Click(object sender, EventArgs e)
    {
        BoomUser.DetailsClient svc = new BoomUser.DetailsClient("DetailsWCF");
```

```
            txtGet_Templates_XML_return.Text = svc.Get_Templates_XML(txtGet_Templates_XML_UsrName.Text,
txtGet_Templates_XML_Passwd.Text);
        }

        private void btnSubmit_Add_Template_Click(object sender, EventArgs e)
        {
            BoomUser.DetailsClient svc = new BoomUser.DetailsClient("DetailsWCF");

            switch
(svc.Add_Template(txtAdd_Template_UsrName.Text,txtAdd_Template_Passwd.Text,txtAdd_Template_Template.Text))
            {
                case BoomUser.Account_Operation_Result.EmptyValue:
                    txtAdd_Template_return.Text = "Empty Value";
                    break;
                case BoomUser.Account_Operation_Result.FieldNotUnique:
                    txtAdd_Template_return.Text = "Field not Unique";
                    break;
                case BoomUser.Account_Operation_Result.InternalError:
                    txtAdd_Template_return.Text = "Internal Error";
                    break;
                case BoomUser.Account_Operation_Result.InvalidAuthentication:
                    txtAdd_Template_return.Text = "Invalid Authentication";
                    break;
                case BoomUser.Account_Operation_Result.InvalidCharacters:
                    txtAdd_Template_return.Text = "Invalid Characters";
                    break;
                case BoomUser.Account_Operation_Result.InvalidDateField:
                    txtAdd_Template_return.Text = "Invalid Date Field";
                    break;
                case BoomUser.Account_Operation_Result.InvalidEmail:
                    txtAdd_Template_return.Text = "Invalid Email";
                    break;
                case BoomUser.Account_Operation_Result.InvalidField:
                    txtAdd_Template_return.Text = "Invalid field";
                    break;
                case BoomUser.Account_Operation_Result.InvalidPartner:
                    txtAdd_Template_return.Text = "Invalid Partner";
                    break;
                case BoomUser.Account_Operation_Result.PasswordMismatch:
                    txtAdd_Template_return.Text = "Password Mismatch";
                    break;
```

```
                case BoomUser.Account_Operation_Result.Success:
                    txtAdd_Template_return.Text = "Success";
                    break;
                default:
                    txtAdd_Template_return.Text = "Unexpected Result";
                    break;
            }

        }

        private void btnSubmit_Remove_Template_Click(object sender, EventArgs e)
        {
            BoomUser.DetailsClient svc = new BoomUser.DetailsClient("DetailsWCF");

            switch (svc.Remove_Template(txtRemove_Template_UsrName.Text, txtRemove_Template_Passwd.Text,
txtRemove_Template_Template.Text))
            {
                case BoomUser.Account_Operation_Result.EmptyValue:
                    txtRemove_Template_return.Text = "Empty Value";
                    break;
                case BoomUser.Account_Operation_Result.FieldNotUnique:
                    txtRemove_Template_return.Text = "Field not Unique";
                    break;
                case BoomUser.Account_Operation_Result.InternalError:
                    txtRemove_Template_return.Text = "Internal Error";
                    break;
                case BoomUser.Account_Operation_Result.InvalidAuthentication:
                    txtRemove_Template_return.Text = "Invalid Authentication";
                    break;
                case BoomUser.Account_Operation_Result.InvalidCharacters:
                    txtRemove_Template_return.Text = "Invalid Characters";
                    break;
                case BoomUser.Account_Operation_Result.InvalidDateField:
                    txtRemove_Template_return.Text = "Invalid Date Field";
                    break;
                case BoomUser.Account_Operation_Result.InvalidEmail:
                    txtRemove_Template_return.Text = "Invalid Email";
                    break;
                case BoomUser.Account_Operation_Result.InvalidField:
                    txtRemove_Template_return.Text = "Invalid field";
                    break;
                case BoomUser.Account_Operation_Result.InvalidPartner:
```

```
                txtRemove_Template_return.Text = "Invalid Partner";
                break;
            case BoomUser.Account_Operation_Result.PasswordMismatch:
                txtRemove_Template_return.Text = "Password Mismatch";
                break;
            case BoomUser.Account_Operation_Result.Success:
                txtRemove_Template_return.Text = "Success";
                break;
            default:
                txtRemove_Template_return.Text = "Unexpected Result";
                break;
        }
    }
```

9. Setting Up Accounts via HTTP POST method

To set-up new Boom SMS accounts via the HTTP POST method a string of arguments must be sent to the Boom SMS website.
The URL is: "https://boom-sms.co.uk/autosetup.asp"

9.1. Building an argument string

The argument string will consist of a key, a value and a set of rules (if required). Each Key / Value / Rules component will be separated by the pipe character ("|") and each group separated by the following string; "#%#".
When building an argument string there are a number of mandatory groups, which must appear in the string in order for the account to be usable once set-up. The following Key/Value groups must appear in the string:

[AccountID]	The username of the new user. *Max 11*
[Password]	The password of the new user *Max 30*
[ContactEMail]	The email address used to contact the user for validation and other account management purposes. *Max 100*
[ContactFirstName]	The user account manager forename *Max 30*
[ContactLastName]	The user account manager surname *Max 30*
[ContactTel]	The user account manager telephone *Max 30*

The following fields are not mandatory but may also be written to:

[SMSSenderId]	The sender name of the user's SMS's *Max 11*
[CompanyName]	The user's company name *Max 80*
[SecondaryEmail]	A secondary Contact Email Address *Max 100*

The contents of the components are:

1. Key : This relates to where the information will be stored in our database, each key name must be enclosed within square brackets
2. Value : What will be stored in the database
3. Rules : What rules, if any to apply to the value submitted (detailed below)

An example of a group is as follows:

```
[AccountID]|JoeBloggs|UQ,RG|
```

In this example the value 'JoeBloggs' is being written into the field '[AccountID]' and the rules supplied here are 'UQ' and 'RG'. This collection of components creates a group. When appending more groups onto your string the result will look like this:

```
[AccountID]|JoeBloggs|UQ,RG|#%#[Password]|abc123||#%#[SMSSenderId]|JoBlgs||
```

Note that in this example there are no rules defined for the Password and SMSSenderId fields. If a field is sent to the Boom SMS HTTP POST setup system without any rules attached the setup system will decide whether the field in question requires validation and will perform them. The usage of rules for developers allows for an extra level of control, but they are not mandatory. Each rule is separated by a comma.

There are four rules available to the developer:

1. UQ: Tells the system to reject non unique values.
2. RG: Randomly generates a value if the supplied value is empty.
3. DT: Validates a given date
4. EM: Basic e-mail address structure validation

As well as the four rules another level of functionality is available by combining the 'UQ' and 'RG' rules. When sending these together, the system will check if the supplied value is unique and if not will randomly generate a different value instead. If an empty value is supplied a unique, randomly generated value will be used.

Note: it is not valid to combine the 'DT' rule with the 'UQ', 'RG' or 'EM' rules.
Note: it is not valid to combine the 'EM' rule with the 'RG' or 'DT' rules.

9.2 Account ID Setup Considerations

The Account ID has to be unique within the Boom-SMS database, and so it is possible that a desired ID may have already been used by someone else. The rules UQ and RG have been set up to aid in automatic account creation.

Currently you are only likely to use these rules in the AccountID group. The Account ID needs to be unique, so you send the UQ rule. The RG rule can be used to get the Boom system to ensure that you get a valid AccountID, randomly generated if required. The actual AccountID is returned to the program that Posted the Request in the Response Data. . For example, you send AccountID through as `[AccountID]|Fred|UQ|`. The Boom-SMS Server finds that "Fred" is NOT unique and has been used before – you will get back an error. If you had sent through `[AccountID]|Fred|UQ,RG|` then the Boom system would have generated an account id based on what you passed through and returned something like "Fred1"

9.3 Posting the argument string to BOOM SMS

Before sending an argument string to the BOOM SMS website an extra header must be added to the POST, this defines the name of the partner to the system to assign the new user to. The Key name of this must be sent as "PARTNERNAME" and the value must be the exact partner name as appears on the partner's account.

Example: Sending a HTTP POST user setup request via ASP

The above specification for defining and sending a new user's details to the BOOM SMS site can be completed using an ASP script. Basic example:

```
Set objSrvHTTP = Server.CreateObject ("MSXML2.ServerXMLHTTP")
ServerURL = "https://boom-sms.co.uk/AutoSetup.asp"
objSrvHTTP.open "POST",ServerURL,false
objSrvHTTP.setRequestHeader "PARTNERNAME", "TEST"

SetupString = "[AccountID]|JoeBloggs|UQ,RG|#%#"&_
"[Password]|abc123||#%#[ContactEmail]|joebloggs@joebloggs.com|EM|#%#"&_
"[ContactFirstName]|Joe||#%#[ContactLastName]|Bloggs||#%#"&_
"[ContactTel]|01234 567890||"
objSrvHTTP.send SetupString
```

If the response text is required, it can be viewed by adding
`Response.Write(objSrvHTTP.responseText)` after the last line of the above code.

Note that the PARTNERNAME in this example is set to 'TEST' this account may be used to verify argument strings but will not actually create accounts. We would request that you use the TEST account during your initial testing phase.

9.4 Return Values

The HTTP POST will return three values

1. STATUS : 'OK' or 'ERROR'
2. ACCOUNTID : New user's account ID, or blank if ERROR
3. MESSAGE : More details on status

The three values will be separated by line breaks (these will not appear if read in a browser), each will be in the form of "KEY=value". Example:

```
STATUS=OK
ACCOUNTID=JoeBloggs
MESSAGE=Required validation : False
```

The following errors may be received when posting malformed argument strings:

1. Partner details incorrect : partner was not found in the database
2. Partner authorisation failed : partner is not authorised to set-up accounts remotely
3. Invalid field name : field name was not recognised
4. Supplied [fieldname] is not a valid date : When a Rule of 'DT' is sent without a valid date
5. Supplied [fieldname] is not unique : duplicate value already in database
6. No [fieldname] supplied : When an empty value is sent without 'RG'
7. Invalid email address supplied : address does not match basic structure rules
8. Incorrect field name syntax : field does not have brackets or has non alphanumeric characters
9. Account setup error : database read / write failure, contact support