[User Manual for ZONG Corporate Bulk SMS API]

Author	Arsalan Azeem
Date	06/04/2015
Version	V 1.0
Status	-
Document Type	-
Copyright © ZonG CMPak – All rights reserved	

Table of Contents

1 Document Control

1.1 Document History

Version	**Date**	**Author**	**Changes since previous version**
1.0	06/04/2015	Arsalan Azeem	Initial Document

2 API methods

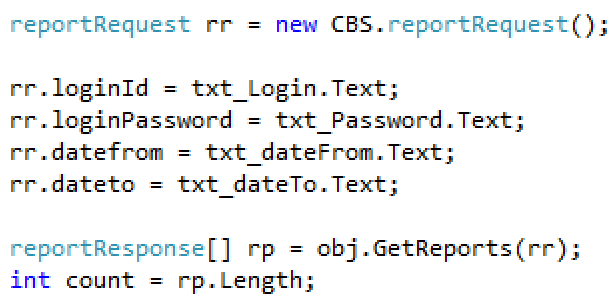
First add the reference of Web API in your solution and make the object of the Service Class. After that classes and methods will be exposed in your code file and you can use them according to the given examples.

2.1 GetReports

This method is used to get the report of the SMS of the specified date and time.

2.1.1 Method Implementation and Detail

Following is screen shot is taken form the code implementation.



```
reportRequest rr = new CBS.reportRequest();

rr.loginId = txt_Login.Text;
rr.loginPassword = txt_Password.Text;
rr.datefrom = txt_dateFrom.Text;
rr.dateto = txt_dateTo.Text;

reportResponse[] rp = obj.GetReports(rr);
int count = rp.Length;
```

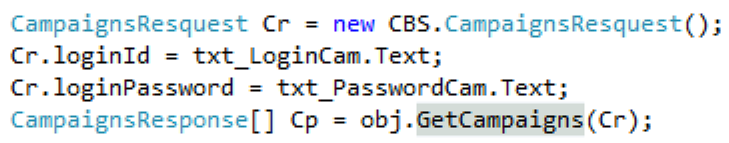
This method accepts the object of the reportRequest class and returns the array of objects of the reportResponse class, as return type is array of object so its give the flexibilityto show this array of objects in any form.
reportRequest class has the property of login id, login password, date form and date to, these are all of the string type.

2.2 GetCampaigns

This method is used to get the all the campaigns of the current user report of the SMS of the specified date and time.

2.2.1 Method Implementation and Detail

Following is screen shot is take form the sample code and the implementation detail.



```
CampaignsResquest Cr = new CBS.CampaignsResquest();
Cr.loginId = txt_LoginCam.Text;
Cr.loginPassword = txt_PasswordCam.Text;
CampaignsResponse[] Cp = obj.GetCampaigns(Cr);
```

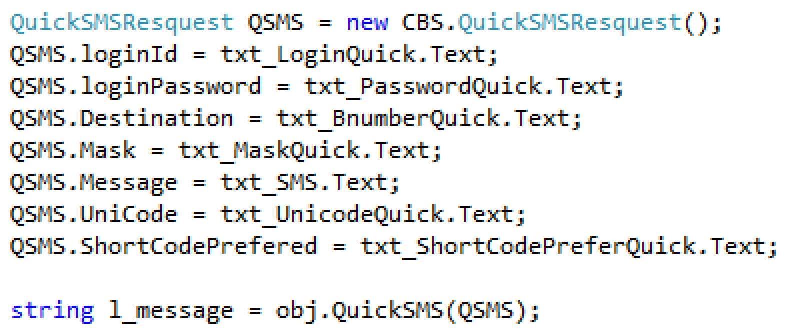
This method accepts the object of the campaignsRequest class and returns the array of objects of the CampaignsResponse class.
reportRequest class has the property of login id and login password, these are all of the string type.

2.3 QuickSMS

This method is used to insert single quick message.

2.3.1 Method Implementation and Detail

Following is screen shot is take form the sample code and the implementation detail.



```
QuickSMSResquest QSMS = new CBS.QuickSMSResquest();
QSMS.loginId = txt_LoginQuick.Text;
QSMS.loginPassword = txt_PasswordQuick.Text;
QSMS.Destination = txt_BnumberQuick.Text;
QSMS.Mask = txt_MaskQuick.Text;
QSMS.Message = txt_SMS.Text;
QSMS.UniCode = txt_UnicodeQuick.Text;
QSMS.ShortCodePrefered = txt_ShortCodePreferQuick.Text;

string l_message = obj.QuickSMS(QSMS);
```

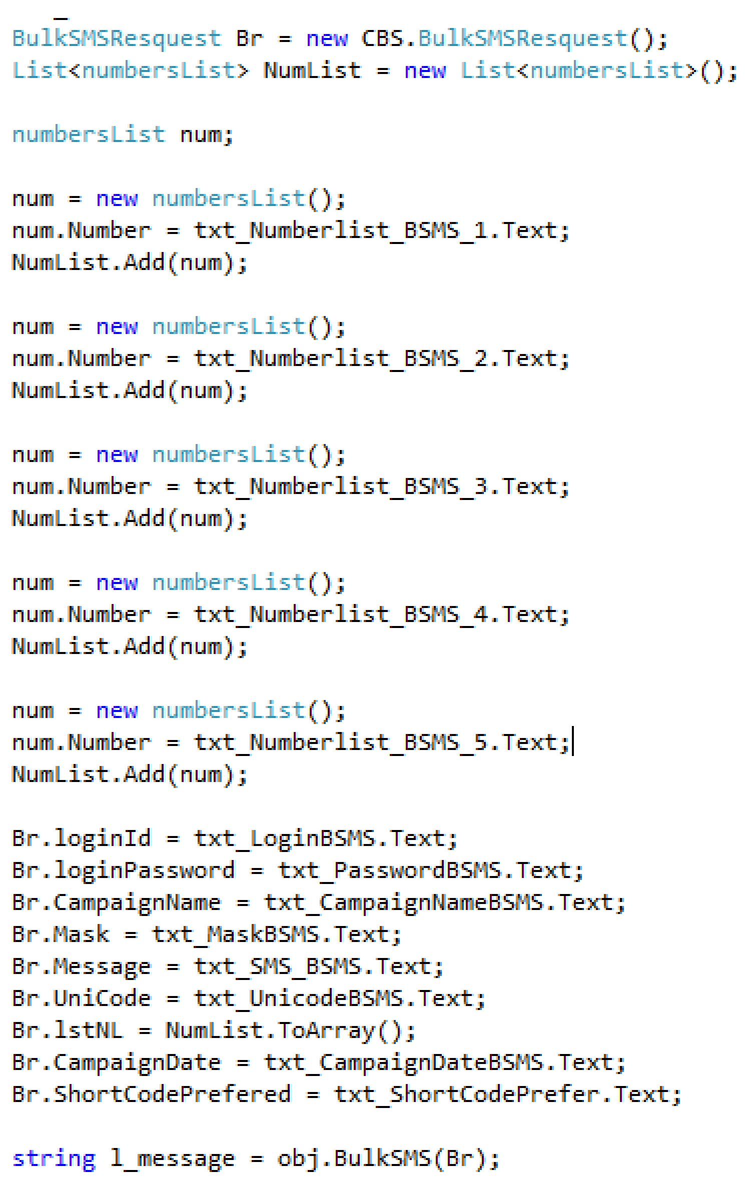
This method accepts the object of the QuickSMSRequest class and returns the string message of success of failure.
QuickSMSRequest class has the property of login id and login password, Destination number (the numebr on which message has to be sent), Uni code (0 for english and 1 for other languages) , shortcodePreferred (this will be used in future so send the default value "n") and the SMS text body, these all properties are of the string type.

2.4 BulkSMS

This method is used to send multiple SMS.

2.4.1 Method Implementation and Detail

Following is screen shot is take form the sample code and the implementation detail.



```
BulkSMSResquest Br = new CBS.BulkSMSResquest();
List<numbersList> NumList = new List<numbersList>();

numbersList num;

num = new numbersList();
num.Number = txt_Numberlist_BSMS_1.Text;
NumList.Add(num);

num = new numbersList();
num.Number = txt_Numberlist_BSMS_2.Text;
NumList.Add(num);

num = new numbersList();
num.Number = txt_Numberlist_BSMS_3.Text;
NumList.Add(num);

num = new numbersList();
num.Number = txt_Numberlist_BSMS_4.Text;
NumList.Add(num);

num = new numbersList();
num.Number = txt_Numberlist_BSMS_5.Text;
NumList.Add(num);

Br.loginId = txt_LoginBSMS.Text;
Br.loginPassword = txt_PasswordBSMS.Text;
Br.CampaignName = txt_CampaignNameBSMS.Text;
Br.Mask = txt_MaskBSMS.Text;
Br.Message = txt_SMS_BSMS.Text;
Br.UniCode = txt_UnicodeBSMS.Text;
Br.lstNL = NumList.ToArray();
Br.CampaignDate = txt_CampaignDateBSMS.Text;
Br.ShortCodePrefered = txt_ShortCodePrefer.Text;

string l_message = obj.BulkSMS(Br);
```

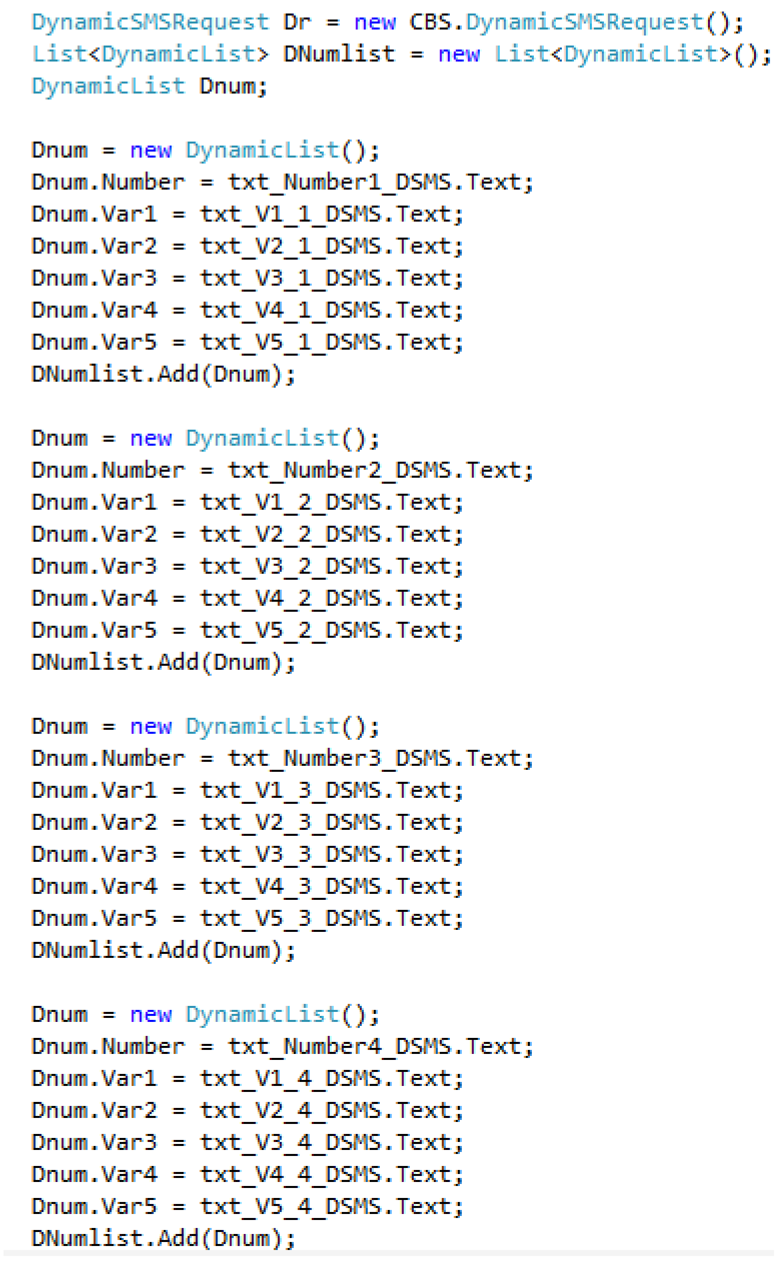
This method accepts the object of the BulkSMSRequest class and returns the string message of success of failure.
BulkSMSRequest class has the property of login id and login password, list/array of Destination numbers (the numebrs on which message has to be sent), Uni code (0 for english and 1 for other languages), campaign date on which sms should sent to the desired numbers, shortcodePreferred (this will be used in future so send the default value "n") and the SMS text body, these all properties are of the string type.

2.5 DynamicSMS

This method is used to send multiple Dynamic SMS.

2.5.1 Method Implementation and Detail

Following is screen shot is take form the sample code and the implementation detail.



```
DynamicSMSRequest Dr = new CBS.DynamicSMSRequest();
List<DynamicList> DNumlist = new List<DynamicList>();
DynamicList Dnum;

Dnum = new DynamicList();
Dnum.Number = txt_Number1_DSMS.Text;
Dnum.Var1 = txt_V1_1_DSMS.Text;
Dnum.Var2 = txt_V2_1_DSMS.Text;
Dnum.Var3 = txt_V3_1_DSMS.Text;
Dnum.Var4 = txt_V4_1_DSMS.Text;
Dnum.Var5 = txt_V5_1_DSMS.Text;
DNumlist.Add(Dnum);

Dnum = new DynamicList();
Dnum.Number = txt_Number2_DSMS.Text;
Dnum.Var1 = txt_V1_2_DSMS.Text;
Dnum.Var2 = txt_V2_2_DSMS.Text;
Dnum.Var3 = txt_V3_2_DSMS.Text;
Dnum.Var4 = txt_V4_2_DSMS.Text;
Dnum.Var5 = txt_V5_2_DSMS.Text;
DNumlist.Add(Dnum);

Dnum = new DynamicList();
Dnum.Number = txt_Number3_DSMS.Text;
Dnum.Var1 = txt_V1_3_DSMS.Text;
Dnum.Var2 = txt_V2_3_DSMS.Text;
Dnum.Var3 = txt_V3_3_DSMS.Text;
Dnum.Var4 = txt_V4_3_DSMS.Text;
Dnum.Var5 = txt_V5_3_DSMS.Text;
DNumlist.Add(Dnum);

Dnum = new DynamicList();
Dnum.Number = txt_Number4_DSMS.Text;
Dnum.Var1 = txt_V1_4_DSMS.Text;
Dnum.Var2 = txt_V2_4_DSMS.Text;
Dnum.Var3 = txt_V3_4_DSMS.Text;
Dnum.Var4 = txt_V4_4_DSMS.Text;
Dnum.Var5 = txt_V5_4_DSMS.Text;
DNumlist.Add(Dnum);
```

Document Title: []	Issue Number:[]	Document Type: ZonG Confidential
Issue Date: []	© ZonG – All rights reserved	Page 7 of 11

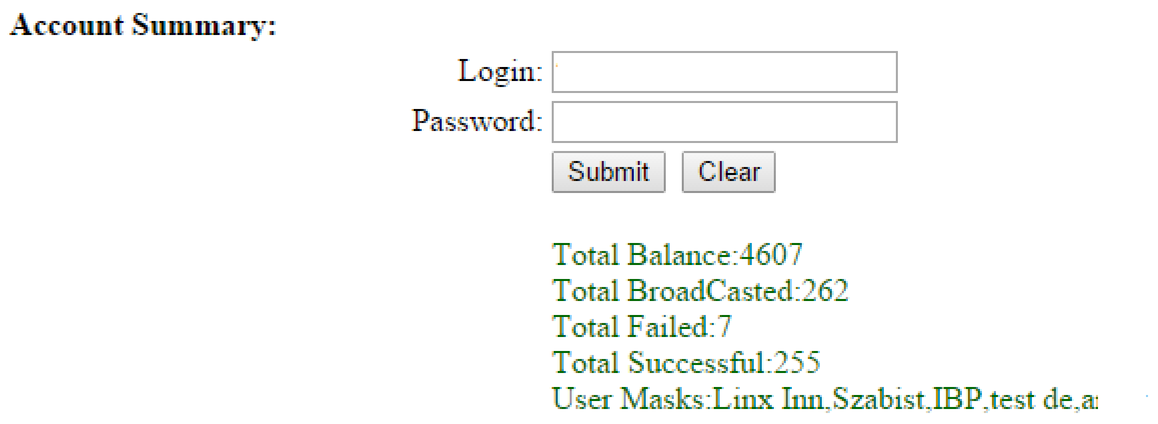
```
Dnum = new DynamicList();
Dnum.Number = txt_Number5_DSMS.Text;
Dnum.Var1 = txt_V1_5_DSMS.Text;
Dnum.Var2 = txt_V2_5_DSMS.Text;
Dnum.Var3 = txt_V3_5_DSMS.Text;
Dnum.Var4 = txt_V4_5_DSMS.Text;
Dnum.Var5 = txt_V5_5_DSMS.Text;
DNumlist.Add(Dnum);

Dr.loginId = txt_Login_DySMS.Text;
Dr.loginPassword = txt_Password_DySMS.Text;
Dr.Mask = txt_Mask_Dy_SMS.Text;
Dr.CampaignDate = txt_CampaignDate_DySMS.Text;
Dr.Message = txt_SMS_DySMS.Text;
Dr.lstNL = DNumlist.ToArray();
Dr.ShortCodePrefered = txt_ShortCodePreferDy.Text;
string l_message = obj.DynamicSMS(Dr);
```

This method takes the destination numbers and the values against each number which will be replaced in the SMS text with provided variables, this method also take the credentials of the account and the campaign date (on which sms has to be sent), Dynamic Message text, masking by which SMS will be sent and shortcodePreferred (this will be used in future so send the default value "n").

2.6 AccountSummary

This method is used to get the status of the account. Output of the Account Summary is like


Account Summary:

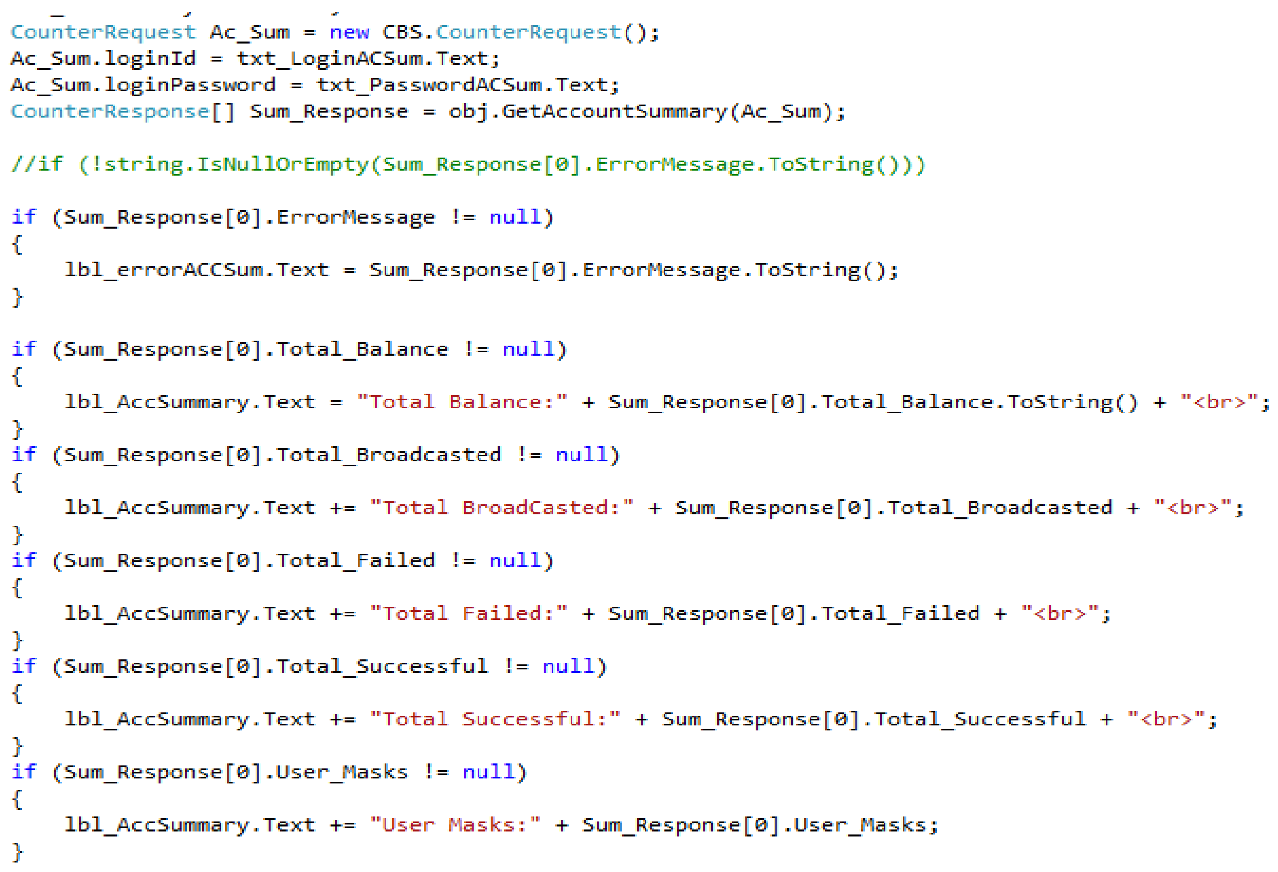
Login:

Password:

Total Balance:4607
Total BroadCasted:262
Total Failed:7
Total Successful:255
User Masks:Linx Inn,Szabist,IBP,test de,a

2.6.1 Method Implementation and Detail

Following is screen shot is take form the sample code and the implementation detail.



```
CounterRequest Ac_Sum = new CBS.CounterRequest();
Ac_Sum.loginId = txt_LoginACSum.Text;
Ac_Sum.loginPassword = txt_PasswordACSum.Text;
CounterResponse[] Sum_Response = obj.GetAccountSummary(Ac_Sum);

//if (!string.IsNullOrEmpty(Sum_Response[0].ErrorMessage.ToString()))

if (Sum_Response[0].ErrorMessage != null)
{
    lbl_errorACCSum.Text = Sum_Response[0].ErrorMessage.ToString();
}

if (Sum_Response[0].Total_Balance != null)
{
    lbl_AccSummary.Text = "Total Balance:" + Sum_Response[0].Total_Balance.ToString() + "<br>";
}
if (Sum_Response[0].Total_Broadcasted != null)
{
    lbl_AccSummary.Text += "Total BroadCasted:" + Sum_Response[0].Total_Broadcasted + "<br>";
}
if (Sum_Response[0].Total_Failed != null)
{
    lbl_AccSummary.Text += "Total Failed:" + Sum_Response[0].Total_Failed + "<br>";
}
if (Sum_Response[0].Total_Successful != null)
{
    lbl_AccSummary.Text += "Total Successful:" + Sum_Response[0].Total_Successful + "<br>";
}
if (Sum_Response[0].User_Masks != null)
{
    lbl_AccSummary.Text += "User Masks:" + Sum_Response[0].User_Masks;
}
```

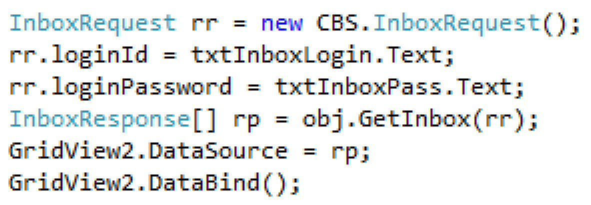
This method accepts the object of CounterRequest Class, which takes only the account credentials in the form of counterRequest class object.

2.7 Inbox

This method is used to get inbox, which is a two way sms functionality provided to limited customers.

2.7.1 Method Implementation and Detail

Following is screen shot is take form the sample code and the implementation detail.



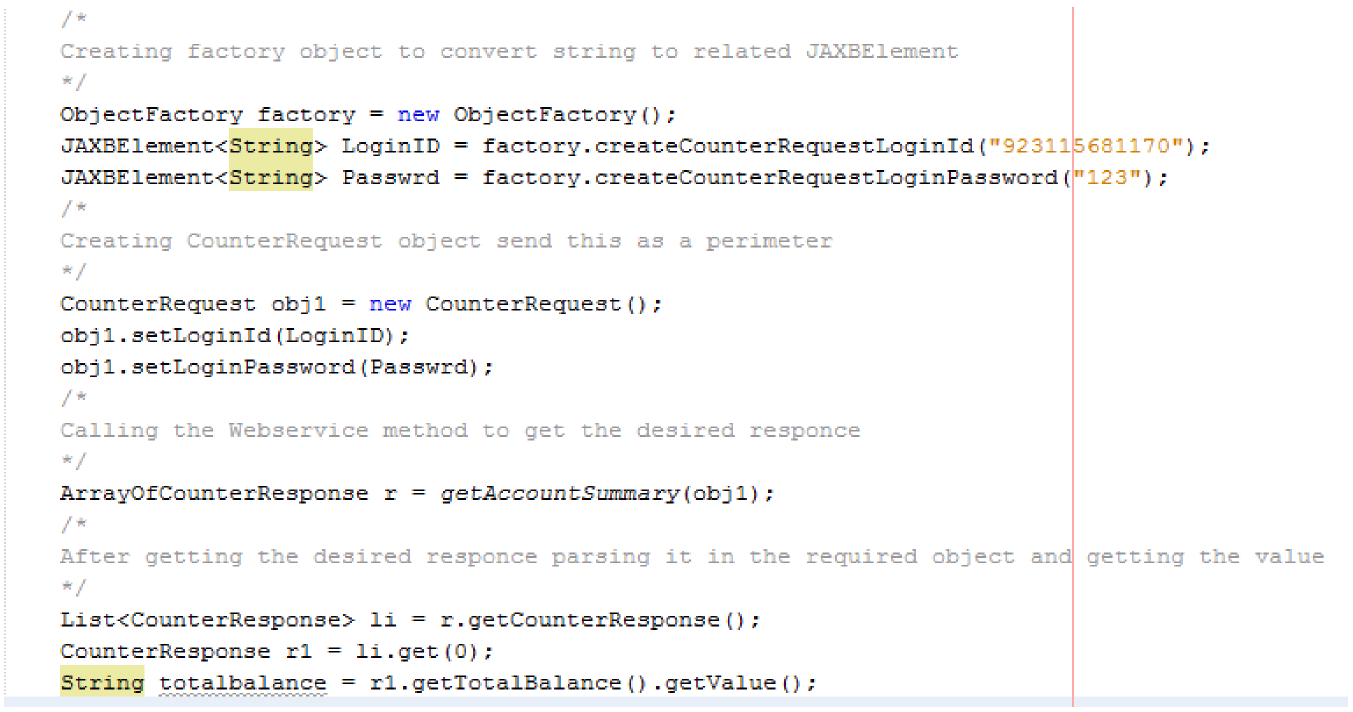
```
InboxRequest rr = new CBS.InboxRequest();
rr.loginId = txtInboxLogin.Text;
rr.loginPassword = txtInboxPass.Text;
InboxResponse[] rp = obj.GetInbox(rr);
GridView2.DataSource = rp;
GridView2.DataBind();
```

This method accepts the object of the InboxRequest class and returns array object of InboxResponse class.
InboxResponse has properties of MobileNo,Reply,ReplyDate and Error Message.

Document Title: []	Issue Number:[]	Document Type: ZonG Confidential
Issue Date: []	© ZonG – All rights reserved	Page 9 of 11

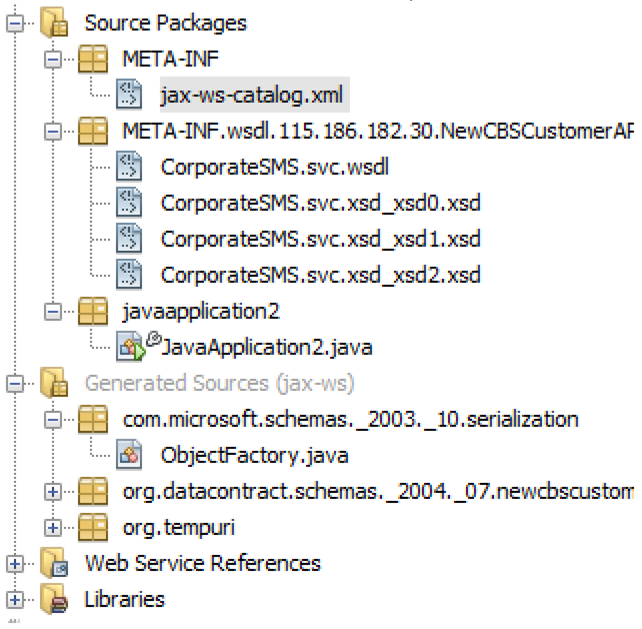
3 JAVA SAMPLE

This is the snapshot of the Client implemented in java



```
/*
Creating factory object to convert string to related JAXBElement
*/
ObjectFactory factory = new ObjectFactory();
JAXBElement<String> LoginID = factory.createCounterRequestLoginId("923115681170");
JAXBElement<String> Passwrd = factory.createCounterRequestLoginPassword("123");
/*
Creating CounterRequest object send this as a perimeter
*/
CounterRequest obj1 = new CounterRequest();
obj1.setLoginId(LoginID);
obj1.setLoginPassword(Passwrd);
/*
Calling the Webservice method to get the desired responce
*/
ArrayOfCounterResponse r = getAccountSummary(obj1);
/*
After getting the desired responce parsing it in the required object and getting the value
*/
List<CounterResponse> li = r.getCounterResponse();
CounterResponse r1 = li.get(0);
String totalbalance = r1.getTotalBalance().getValue();
```

This is the folder structure of the java client created

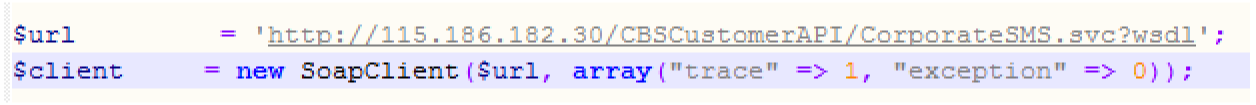


```
Source Packages
    META-INF
        jax-ws-catalog.xml
    META-INF.wsdl.115.186.182.30.NewCBSCustomerAF
        CorporateSMS.svc.wsdl
        CorporateSMS.svc.xsd_xsd0.xsd
        CorporateSMS.svc.xsd_xsd1.xsd
        CorporateSMS.svc.xsd_xsd2.xsd
    javaapplication2
        JavaApplication2.java
Generated Sources (jax-ws)
    com.microsoft.schemas._2003._10.serialization
        ObjectFactory.java
    org.datacontract.schemas._2004._07.newcbscustom
    org.tempuri
Web Service References
Libraries
```

4 PHP SAMPLE

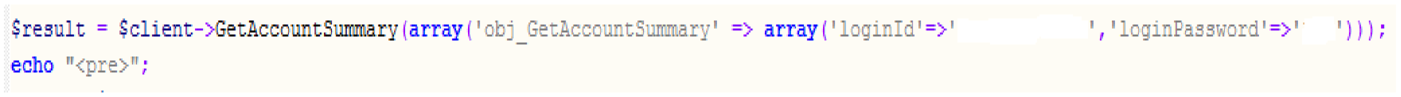
Two Method of API were descirbed below for PHP code.

API reference:



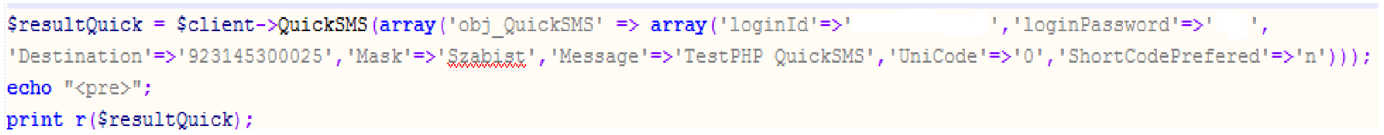
```
$url            = 'http://115.186.182.30/CBSCustomerAPI/CorporateSMS.svc?wsdl';
$client       = new SoapClient($url, array("trace" => 1, "exception" => 0));
```

1. **GetAccountSummary Mehtod Implementation:**



```
$result = $client->GetAccountSummary(array('obj_GetAccountSummary' => array('loginId'=>'            ','loginPassword'=>'     ')));
echo "<pre>";
```

2. **Quick SMS Mehtod Implementation:**



```
$resultQuick = $client->QuickSMS(array('obj_QuickSMS' => array('loginId'=>'            ','loginPassword'=>'    ',
'Destination'=>'923145300025','Mask'=>'Szabist','Message'=>'TestPHP QuickSMS','UniCode'=>'0','ShortCodePrefered'=>'n')));
echo "<pre>";
print r($resultQuick);
```