



## **SMS Gateway (SOAP/XML Web Service)**

### **Technical Documentation**

**Last Updated: 11/02/2004**

**To use the SOAP/XML Web Service functionality you will need a valid account.**

**For information about obtaining an account please see our web site at**

**[developer.bondwireless.com](http://developer.bondwireless.com)**

**Please contact [support@bondwireless.com](mailto:support@bondwireless.com) with any queries.**

# Contents

Soap Technical Specifications .....	3
Server Throughput .....	3
Connection Details .....	3
SMSGateway Messaging Service Endpoint .....	3
WSDL File .....	3
Methods .....	3
Sending SMS .....	3
Receiving SMS .....	5
Example Code .....	8
MS .NET (VB) Sample Code .....	8
MS ASP/VB6 Sample Code .....	8
Java Code .....	9

## Soap Technical Specifications

### ***Server Throughput***

Running on the current hardware, the Bond Wireless SMS Gateway can handle over 3000 requests from SOAP clients a minute.

Our SMS throughput is limited by the carriers we connect to. Currently most of our carriers can handle a rate of up to 500 messages per second, depending on current SMSC load and destination country/network of the message.

Incoming SMS throughput depends on the network of origin, but generally our current platform can accept up to 500 messages a second. Processing times vary based on the application.

### ***Connection Details***

#### **SMSGateway Messaging Service Endpoint**

<http://soap.bondwireless.com/soap/services/SmsGateway>

#### **WSDL File**

<http://soap.bondwireless.com/soap/services/SmsGateway?wsdl>

#### **Methods**

There are many methods visible in the WSDL; however, currently the following are the most applicable to developers.

#### **Sending SMS**

int **SendSms** (String to, String message, String userId, String password)

*Primary method for sending SMS messages. As no CLI (originator) is specified, the message will come from the default CLI in the user account preferences. The account will be charged at 1 credit per message for most networks. Some networks are charged at more than 1 credit. See our coverage list for details.*

*“to” is the destination number in international format e.g. +61403000000.*

*“message” is the content of your SMS. There is no character length restriction, but you will be charged 1 credit per 160 characters.*

*“userId” and “password” are your Bond Wireless developer account details.*

*Return value of function is an error/status code – this code can be looked up using the function ErrorMessage(int errorCode).*

int **SendSmsFrom**(String to, String from, String message, String userId, String password)

*Send an SMS, with a given CLI (originator) (“from”). CLI must be no longer than 11 characters or 16 numbers.*

int **SendSmsWithGateway** (String to, String from, String message, String userId, String password, int gateway)

*Send SMS through a specific Bond Wireless Gateway, different to the one setup on your account. Contact support for more information.*

int **QueueSmsFrom**(String to, String from, String message, String userId, String password, long minutes)

*Queue a message, to be sent in a given number of minutes.*

int **SendSmsMultiDestination**(String to, String from, String message, String userId, String password)

*Send same message to multiple numbers. Separate numbers using “;” in the “to” field.*

int **QueueSmsMultiDestination**(String to, String from, String message, String userId, String password, Long minutes)

*Send same message to multiple numbers in a given number of minutes.*

int **SendMultiPersonalizedMultiDestination**(String to, String from, String message, String userId, String password, String fields, String[] personalizations)

*Send SMS to multiple destinations with multiple custom fields. Fields must be in the format of “{fieldname}”. They can be anywhere in your message, but need to be specified as a semi-colon separated string for the “fields” parameter. The “personalizations” array contains the fields for each destination in order, also separated by semi-colon.*

*Example:*

To = “+61402007007;+61402008008”

From = “JoesGarage”

Message = “Hi {first-name}, your car will be ready at {time}. The total bill is {cost}.”

Fields = “{firstname};{time};{cost}”

Personalizations[] = [{"Nick;4:30pm;\$8538.00"}, {"Gareth;5:00pm;\$108.00"}]

SendMultiPersonalizedMultiDestination(To,From,Message,”testaccount”,”password”,Fields,Personalizations[])

Will send these messages;

1. To: +61402007007: Hi Nick, your car will be ready at 4:30pm. The total bill is \$8538.00

And

- To: +61402008008: Hi Gareth, your car will be ready at 5:00pm. The total bill is \$108.00

int **QueueMultiPersonalizedMultiDestination**(String to, String from, String message, String userId, String password, String fields, String[] personalizations, Long minutes)

*As above, but in a given number of minutes.*

## **Receiving SMS**

Prior to using this functionality, your account must have an incoming service applied to it. Please contact your Account Manager to add this service.

Different programming languages handle arrays in XML Web Services differently, so two options are given for each of these methods. Functionality is the same in either option.

String[][] **GetSmsReplyCampaigns**(String userId, String password)

*Returns a two-dimensional array of the incoming SMS services associated with your account. The fields returned are MSIDN and CampaignID. The MSIDN is the actual number associated with the incoming service, and the CampaignID is a unique identifier used to associate your account with the incoming service. (Multiple accounts may share the same MSIDN in some instances).*

*Eg: {{ "a","b" }, {"a","b" }}, where a = MSIDN; b= CampaignID*

String[] **GetSmsReplyCampaignsList**(String userId, String password)

*Returns a one-dimensional array of the incoming SMS services associated with your account. The MSIDN and CampaignID fields are separated by “|” within each cell of the array.*

String[][] **GetSmsReplies**(String campaignId, String userId, String password)

*Returns a two-dimensional array of any new, 'unread' incoming SMS messages. The fields returned are Message, Sender and Timestamp.*

*Eg: {{ "a","b","c" }, {"a","b","c" }}, where a = message; b= sender; c = timestamp  
Note that the timestamp is in AEST.*

String[] **GetSmsRepliesList**(String campaignId, String userId, String password)

*Returns a one-dimensional array of your new incoming SMS messages. The Message, Sender and Timestamp fields are separated by “|” within each cell of the array.*

## **Administration Methods**

String **ErrorMessage**(int errorCode)

*Look up the description of an error message associated with an error/status code.*

int **CreditsAvailable** (String userId, String, String userPassword)

*Available credits on the given account.*

int **SetUserPassword**(String userId, String newPassword, String password)

*Change a users password.*

String **GetUserDefaultCLI**(String userId, String password)

*Return the default CLI for a user.*

int **SetUserDefaultCLI**(String CLI, String userId, String password)

*Set the default CLI for a user.*

int **CreateSubUser** (String newUserId, String newPassword, String adminUserId, String adminPassword)

*Create a subaccount from your parent account (adminUserId and adminPassword). (Zero credits will be allocated).*

int **CreateSubUserWithCredits**(String newUserId, String newPassword, int credits, String adminUserId, String adminPassword)

*Create a subaccount from your parent account (adminUserId and adminPassword), with a given number of credits (taken from the parent account). If you specify credits as "-1", the user will have unlimited credits, deducting from the parent account as they are used.*

int **AddCreditsSubUser**(String subAccount, int credits, String adminUserId, String adminPassword)

*Add credits from a parent account to a sub account. adminUserId and adminPassword are the account details for the parent account.*

Int **DeleteCreditsSubUser**(String subAccount, int credits, String adminUserId, String adminPassword)

*Remove credits from a sub account to a parent account. adminUserId and adminPassword are the account details for the parent account.*

int **DeleteSubUser** (String userId, String adminUserId, String adminPassword)  
*Delete a sub account. (Any remaining credits will be moved back to the parent).*

String[] **ListSubAccounts** (String userId, String userPassword)  
*Returns an array of all sub accounts under the given account, including grandchildren.*

int **SubAccountsCreditsAvailable**(String userId, String, String adminUserId, String adminPassword)  
*Available credits on sub account. adminUserId and adminPassword are the account details for the parent account.*

## **Example Code**

### **MS .NET (VB) Sample Code**

1. Go to “Project” menu, choose “Add Web Reference” enter our XML Web Service WSDL URL: <http://soap.bondwireless.com/soap/services/SmsGateway?wsdl>

2. Add the following code to your class:

```
Imports System.IO

Dim soapService As New com.bondwireless.soap.SmsGatewayService()
```

The SOAP methods are now available in your .net development environment:

```
Eg. soapService.SendSmsFrom(Number, From, Message, UserId,
Password)
```

For more information, see:

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vsintro7/html/vxtskAddingRemovingWebReferences.asp>

### **MS ASP/VB6 Sample Code**

This method of connection can be used for older clients not using the .NET framework. In order to use the following code you will require Microsoft’s SOAP toolkit to be installed on the client machine.

```
Const BASE_SOAP_ACTION_URI =
"http://soap.bondwireless.com/soap/services/SmsGateway"
Const WRAPPER_ELEMENT_NAMESPACE =
"http://soap.bondwireless.com/soap/services/SmsGateway"

Const END_POINT_URL =
"http://soap.bondwireless.com/soap/services/SmsGateway"

Dim Answer
Dim Operation
Dim Serializer
Dim Reader
Dim Connector

Dim ToNum
ToNum = Request.Form("to")
FromText = Request.Form("from")
Message = Request.Form("message")
UserId = Request.Form("userId")
Password = Request.Form("password")

Operation = "SendSmsFrom"
```

```

Set Serializer = CreateObject("MSSOAP.SoapSerializer")

Set Connector = CreateObject("MSSOAP.HttpConnector")
Connector.Property("EndPointURL") = END_POINT_URL

Connector.Property("SoapAction") = BASE_SOAP_ACTION_URI & Operation

Connector.BeginMessage

Set Serializer = CreateObject("MSSOAP.SoapSerializer")
Serializer.Init Connector.InputStream

Serializer.startEnvelope
Serializer.startBody
Serializer.startElement Operation, WRAPPER_ELEMENT_NAMESPACE, "m"
Serializer.startElement "to"
Serializer.writeString ToNum
Serializer.endElement
Serializer.startElement "from"
Serializer.writeString FromText
Serializer.endElement
Serializer.startElement "message"
Serializer.writeString Message
Serializer.endElement
Serializer.startElement "userId"
Serializer.writeString UserId
Serializer.endElement
Serializer.startElement "password"
Serializer.writeString Password
Serializer.endElement
Serializer.endElement
Serializer.endBody
Serializer.endEnvelope

Connector.EndMessage

Set Reader = CreateObject("MSSOAP.SoapReader")
Reader.Load Connector.OutputStream

If Reader.Fault Is Nothing Then
    Answer = Reader.RPCResult.Text
Else
    Answer = Reader.faultstring.Text
End If

```

## Java Code

For information about connecting to the Bond Wireless SMS Web Service using Java, please contact our Technical Department for the SDK.