



Web Services API

Reference Manual

Date Commenced: 3rd August 2006

Last Modified: 6th March 2007

Version: 1.2



Table of Contents

Overview.....	3
Web Methods.....	4
Authentication	4
SubmitJob	4
Job.....	4
FaxJob.....	4
EmailJob.....	5
SmsJob.....	5
TtsJob.....	6
VoiceJob.....	6
Sender	7
File.....	7
Recipient.....	8
PhoneRecipient	8
FaxRecipient	8
EmailRecipient	9
SmsRecipient	9
TtsRecipient	9
VoiceRecipient	10
MergeField	10
RetrieveReport.....	11
RetrieveDetailedReport.....	11
StatusReport	11
DetailedReport	12
ReportRecipient.....	12
Using Merge Fields.....	13
Code Samples	14
C# Examples.....	14
Java Axis Examples	22
Appendix.....	32
Valid number formats.....	32
Broadcast Status Codes	32
Send Codes	33



Overview

The OzMedia Web Service API allows a registered user to submit a variety of messages and retrieve reports using SOAP over HTTP.

The location of the Web Service, service description and WSDL can be found at

<http://users.ozmedia.com/webservice/service.asmx>

This document is technical in nature and it is assumed that the reader is familiar with SOAP and the implementation of web services.

A note on code samples: These have been provided in .Net c# and Java using callable methods; however, any high level SOAP toolkit should provide the same functionality. The web service is cross platform and conforms to the web service standards.



Web Methods

Authentication

All web methods require an authentication header consisting of username and password to be submitted within the SOAP header.

Submit Job

This method is used to submit a Fax, Email, SMS, Text to Speech or Voice job to the OzMedia messaging system.

Expects:

Job entity conforming to one of the following types:- FaxJob, EmailJob, SmsJob, TtsJob or VoiceJob

Returns:

Integer representing the job id or an exception on error.

Job

Job is an abstract class – submitted jobs must be one of FaxJob, EmailJob, SmsJob, TtsJob or VoiceJob.

Properties:

Property	Type	Required	Description
Name	String	Yes	A user defined name for the job
Scheduled	DateTime	No	A future date and time for the job to be activated
ListIds[]	Integer	No*	An array of list id's if using lists already uploaded to the OzMedia website.
Sender	Entity	No	Complex type consisting of sender information.
Recipients[]	Recipient	No*	Array of message recipients.

- One or more ListIds OR one or more Recipient required. Note that you may not use a combination of both ListIds AND Recipient as the object model differs between the two.

FaxJob

Extends **Job**

Properties:

Property	Type	Required	Description
Name	String	Yes	A user defined name for the job
Scheduled	DateTime	No	A future date and time for the job to be activated
ListIds[]	Integer	No*	An array of list id's if using lists already uploaded to the OzMedia website.
Sender	Entity	No	Complex type consisting of sender information.
Files[]	File	Yes	One or more file objects
Recipients[]	FaxRecipient	No	Required if no ListIds submitted. FaxRecipient derives from Recipient



EmailJob

Extends **Job**

Properties:

Property	Type	Required	Description
Name	String	Yes	A user defined name for the job
Scheduled	DateTime	No	A future date and time for the job to be activated
ListIds[]	Integer	No*	An array of list id's if using lists already uploaded to the OzMedia website.
Sender	Entity	No	Complex type consisting of sender information.
Subject	String	Yes	Email subject
Text	String	No**	Text version of email (required if no HTML version exists)
Html	String	No**	HTML version of email (required if no Text version exists)
Files[]	File	No	One or more file objects to be attached to email
Recipients[]	EmailRecipient	No	Required if no ListIds submitted. EmailRecipient derives from Recipient

** Either one or both of Text, Html is required

SmsJob

Extends **Job**

Properties:

Property	Type	Required	Description
Name	String	Yes	A user defined name for the job
Scheduled	DateTime	No	A future date and time for the job to be activated
ListIds[]	Integer	No*	An array of list id's if using lists already uploaded to the OzMedia website.
Sender	Entity	No	Complex type consisting of sender information.
Text	String	Yes	SMS text. Restriction of 458 characters.
IsTwoWay	Boolean	Yes	Is this a two way SMS job?
ExpiryHours	Byte	No	If this is a two way SMS job, how many hours before job should expire?
BatchReplies	Boolean	Yes	Should replies be batched at expiry? If not, replies will be sent individually to Sender.ReplyTo value as they arrive. For Two Way Jobs only.
Recipients[]	SmsRecipient	No	Required if no ListIds submitted. SmsRecipient derives from Recipient



TtsJob

Extends **Job**

Properties:

Property	Type	Required	Description
Name	String	Yes	A user defined name for the job
Scheduled	DateTime	No	A future date and time for the job to be activated
ListIds[]	Integer	No*	An array of list id's if using lists already uploaded to the OzMedia website.
Sender	Entity	No	Complex type consisting of sender information.
Text	String	Yes	Text to speech text.
IsTwoWay	Boolean	Yes	Is this a two way TTS job?
KeypressOnly	Boolean	Yes	If the purpose of your job is to collect keypresses only and not transfer to a third party, set to true. Note: you must also set IsTwoWay to true for keypresses to be collected
SuppressHeader	Boolean	Yes	Suppress the introductory header?
PreferredDestinationType	Enum	No	UseDefault = 0 UseLandline = 1 UseMobile = 2
TransferKeys[]	Entity	No	Required for Two Way TTS jobs (not required where KeypressOnly = true)
Recipients[]	TtsRecipient	No	Required if no ListIds submitted. TtsRecipient derives from Recipient

VoiceJob

Extends **Job**

Properties:

Property	Type	Required	Description
Name	String	Yes	A user defined name for the job
Scheduled	DateTime	No	A future date and time for the job to be activated
ListIds[]	Integer	No*	An array of list id's if using lists already uploaded to the OzMedia website.
Sender	Entity	No	Complex type consisting of sender information.
Recording	File	Yes	File object – only WAV files are accepted.
Recipients[]	VoiceRecipient	No	Required if no ListIds submitted. VoiceRecipient derives from Recipient



Sender

Properties:

Property	Type	Required	Description
Name	String	No	Contact name of user submitting job. If not provided, the company name from the stored user profile will be used.
Company	String	No	Company name. If not provided, the company name from the stored user profile will be used. Company name will be mentioned in TTS broadcasts as part of the introductory message
ReplyTo	String	No	Reply email (for email job) or number (for SMS job). If not provided, details in stored user profile will be used. The reply email will be displayed in email broadcasts; the reply mobile number will be displayed in one way SMS jobs.

File

Properties:

Property	Type	Required	Description
Name	String	Yes	File name including extension
Priority	Integer	No	Priority order for multiple files
Content[]	Byte	Yes	Byte array of file content

Recipient

An abstract class representing a message recipient.

Properties:

Property	Type	Required	Description
Reference	String	No	User defined reference for this recipient
Title	String	No	Recipient title (e.g. Mr, Ms etc)
FirstName	String	No	First name of recipient
LastName	String	No	Last name of recipient
Destination	String	Yes	Destination – e.g. fax number, email address, phone number or mobile number.

PhoneRecipient

An abstract class representing a phone recipient. Extends **Recipient**

Properties:

Property	Type	Required	Description
Reference	String	No	User defined reference for this recipient
Title	String	No	Recipient title (e.g. Mr, Ms etc)
FirstName	String	No	First name of recipient
LastName	String	No	Last name of recipient
Destination	String	Yes	Valid phone, fax or mobile number

Valid number construction for Destination:-

Local number with area code	e.g. 02 9123 4567
International number with “+”	e.g. +44 2 1234 5678
International number with “0011”	e.g. 0011 44 2 1234 5678

FaxRecipient

Extends PhoneRecipient.

Properties:

Property	Type	Required	Description
Reference	String	No	User defined reference for this recipient
Title	String	No	Recipient title (e.g. Mr, Ms etc)
FirstName	String	No	First name of recipient
LastName	String	No	Last name of recipient
Destination	String	Yes	Valid phone, fax or mobile number



EmailRecipient

Extends **Recipient**

Properties:

Property	Type	Required	Description
Reference	String	No	User defined reference for this recipient
Title	String	No	Recipient title (e.g. Mr, Ms etc)
FirstName	String	No	First name of recipient
LastName	String	No	Last name of recipient
Destination	String	Yes	Valid email address
MergeField[]	Entity	No	A list of merge key/value pairs

Valid construction for Destination:- any valid email address

SmsRecipient

Extends **PhoneRecipient**

Properties:

Property	Type	Required	Description
Reference	String	No	User defined reference for this recipient
Title	String	No	Recipient title (e.g. Mr, Ms etc)
FirstName	String	No	First name of recipient
LastName	String	No	Last name of recipient
Destination	String	Yes	Valid phone style number
MergeField[]	Entity	No	A list of merge key/value pairs

TtsRecipient

Extends **PhoneRecipient**

Properties:

Property	Type	Required	Description
Reference	String	No	User defined reference for this recipient
Title	String	No	Recipient title (e.g. Mr, Ms etc)
FirstName	String	No	First name of recipient
LastName	String	No	Last name of recipient
Destination	String	Yes	Valid phone style number
MergeField[]	Entity	No	A list of merge key/value pairs



VoiceRecipient

Extends **PhoneRecipient**

Properties:

Property	Type	Required	Description
Reference	String	No	User defined reference for this recipient
Title	String	No	Recipient title (e.g. Mr, Ms etc)
FirstName	String	No	First name of recipient
LastName	String	No	Last name of recipient
Destination	String	Yes	Valid phone style number

MergeField

Properties:

Property	Type	Required	Description
Key	String	Yes	Key relating to the merge field in message to be replaced.
Value	String	Yes	Value to replace key with for specific recipient.



RetrieveReport

Use this method to retrieve information about a specific OzMedia job by Job ID.

Expects:

Integer **jobId** representing the job to be retrieved.

Returns:

StatusReport entity

RetrieveDetailedReport

Use this method to retrieve a detailed job report by Job ID.

Expects:

Integer **jobId** representing the job to be retrieved.

Returns:

DetailedReport entity

StatusReport

Entity representing the status report of a job.

Properties:

Property	Type	Description
JobId	Integer	The job id to be retrieved.
Submitted	Boolean	A summary report will return summary job information without recipient list. By setting to false you will retrieve the status list of all recipients if a job is in progress or complete.
JobType	String	FAX, Email, Voice, SMS, Text-To-Speech
JobStatus	String	Scheduled, Pending, Broadcast Submitted, Complete, Closed, Queued
TotalRecipients	Integer	Total number of recipients submitted
SentRecipients	Integer	Total number successfully sent to
FailedRecipients	Integer	Total number of failed recipients
TotalCost	Double	Total cost of job (if complete or closed)



DetailedReport

Extends **StatusReport**

Properties:

Property	Type	Description
JobId	Integer	The job id to be retrieved.
Submitted	Boolean	A summary report will return summary job information without recipient list. By setting to false you will retrieve the status list of all recipients if a job is in progress or complete.
JobType	String	FAX, Email, Voice, SMS, Text-To-Speech
JobStatus	String	Scheduled, Pending, Broadcast Submitted, Complete, Closed, Queued
TotalRecipients	Integer	Total number of recipients submitted
SentRecipients	Integer	Total number successfully sent to
FailedRecipients	Integer	Total number of failed recipients
TotalCost	Double	Total cost of job (if complete or closed)
ReportRecipient[]	Entity	Array of completed recipients for the job

ReportRecipient

Properties:

Property	Type	Description
Recipient	String	Recipient name
Reference	String	Recipient reference
Destination	String	For user lists only:- fax, phone or email
Status	String	SENT, ERR etc.
Duration	Integer	Fax/TTS/voice jobs only – duration of message
Cost	Double	Cost of message
Keypress	Integer	2 Way TTS jobs only – keypress user selected
Reply	String	2 Way SMS jobs only – user reply



Using Merge Fields

Merge fields may be used in the following jobs only:-

1. Email
2. SMS
3. Text-To-Speech

To indicate a merge field in your message, the field should be enclosed by double percentage characters – e.g. %%MyField%%

There are reserved merge fields which you may use without specifically creating using the merge key/value pairs. These are:

1. %%Title%% - recipient's title
2. %%FirstName%% - recipient's first name
3. %%LastName%% - recipient's last name
4. %%Recipient%% - will be replaced by recipient's full name
5. %%Reference%% - will be replaced by the recipient's reference
6. %%Email%% - will be replaced by the recipient's destination for Email type jobs
7. %%Mobile%% - will be replaced by the recipient's destination for SMS type jobs
8. %%Phone%% - will be replaced by the recipient's destination for Text-To-Speech type jobs

Sample message:

Attention: %%Recipient%%

Dear %%FirstName%%,

According to our records you are currently overdue on payment for %%BillName%%. The amount owing is %%BillAmount%%.

Please organize payment before the due date of %%DueDate%%.

Sample Construction

```
<SmsRecipient>
  <Title>Mr</Title>
  <FirstName>Andrew</FirstName>
  <LastName>Citizen</LastName>
  <Reference>19462</Reference>
  <Destination>0412 345 678</Destination>
  <MergeField>
    <Key>BillName</Key>
    <Value>June Electricity Bill</Value>
  </MergeField>
  <MergeField>
    <Key>BillAmount</Key>
    <Value>$78.32</Value>
  </MergeField>
  <MergeField>
    <Key>DueDate</Key>
    <Value>1 August 2007</Value>
  </MergeField>
</SmsRecipient>
```



Code Samples

C# Examples

```
public void TestFax()
{
    List<FaxRecipient> recipientList = new List<FaxRecipient>();
    FaxRecipient recipient = new FaxRecipient();
    recipient.Title = "Ms";
    recipient.FirstName = "Annette";
    recipient.LastName = "Citizen";
    recipient.Reference = "Of the world";
    recipient.Destination = "02 9123 4567";

    recipientList.Add(recipient);

    List<File> fileList = new List<File>();

    File myFile = new File();
    myFile.Name = "test.doc";
    myFile.Priority = 1;
    System.IO.FileStream file = new
        System.IO.FileStream("files/test.doc",
            System.IO.FileMode.Open, System.IO.FileAccess.Read);
    System.IO.BinaryReader br = new System.IO.BinaryReader(file);
    byte[] buffer = new byte[(int)file.Length];
    br.Read(buffer, 0, (int)file.Length);
    br.Close();

    myFile.Content = buffer;

    fileList.Add(myFile);

    FaxJob faxJob = new FaxJob();
    faxJob.Name = "Test Fax";
    faxJob.Sender = new Sender();
    faxJob.Sender.Company = "My Company";
    faxJob.Sender.Name = "My Name";
    faxJob.Sender.ReplyTo = "02 9876 5432";

    faxJob.Recipients = recipientList.ToArray();
    faxJob.Files = fileList.ToArray();

    AuthenticationHeader authHeader = new AuthenticationHeader();
    authHeader.Username = "test";
    authHeader.Password = "test";

    WelWebService service = new WelWebService();
    service.AuthenticationHeaderValue = authHeader;
    int jobid = service.SubmitJob(faxJob);
}
```

OZMEDIA

```
public void TestEmail()
{
    List<EmailRecipient> recipientList = new List<EmailRecipient>();

    EmailRecipient recipient = new EmailRecipient();
    recipient.Title = "Ms";
    recipient.FirstName = "Annette";
    recipient.LastName = "Citizen";
    recipient.Reference = "Of the world";
    recipient.Destination = "a.citizen@theworld.com ";
    List<MergeField> fieldList = new List<MergeField>();
    MergeField mergeField = new MergeField();
    mergeField.Key = "FavouriteAnimal";
    mergeField.Value = "Cats";
    fieldList.Add(mergeField);

    recipient.MergeFields = fieldList.ToArray();

    recipientList.Add(recipient);

    List<File> myFiles = new List<File>();
    File myFile = new File();
    myFile.Name = "test.jpg";
    myFile.Priority = 1;
    System.IO.FileStream file = new
        System.IO.FileStream("files/test.jpg",
            System.IO.FileMode.Open, System.IO.FileAccess.Read);
    System.IO.BinaryReader br = new System.IO.BinaryReader(file);
    byte[] buffer = new byte[(int)file.Length];
    br.Read(buffer, 0, (int)file.Length);
    br.Close();

    myFile.Content = buffer;
    myFiles.Add(myFile);

    EmailJob emailJob = new EmailJob();
    emailJob.Name = "Test Email";
    emailJob.Sender = new Sender();
    emailJob.Sender.Company = "My Company";
    emailJob.Sender.Name = "Test Person";
    emailJob.Sender.ReplyTo = "test@test.com";
    emailJob.Subject = "My Test Email";
    emailJob.Recipients = recipientList.ToArray();
    emailJob.Text = "Hi %%FirstName%%, your favourite animal is
        %%FavouriteAnimal%%.";
    emailJob.Files = myFiles.ToArray();

    AuthenticationHeader authHeader = new AuthenticationHeader();
    authHeader.Username = "test";
    authHeader.Password = "test";

    WelWebService service = new WelWebService();
    service.AuthenticationHeaderValue = authHeader;
    int jobid = service.SubmitJob(emailJob);
}
```

OZMEDIA

```
public void TestSMS()
{
    List<SmsRecipient> recipientList = new List<SmsRecipient>();

    SmsRecipient recipient = new SmsRecipient();
    recipient.Title = "Ms";
    recipient.FirstName = "Annette";
    recipient.LastName = "Citizen";
    recipient.Reference = "Of the world";
    recipient.Destination = "+61 4 9876 5432";

    List<MergeField> fieldList = new List<MergeField>();
    MergeField mergeField = new MergeField();
    mergeField.Key = "FavouriteAnimal";
    mergeField.Value = "Cats";
    fieldList.Add(mergeField);

    recipient.MergeFields = fieldList.ToArray();
    recipientList.Add(recipient);

    SmsJob smsJob = new SmsJob();
    smsJob.Name = "SMS Test";
    smsJob.Sender = new Sender();
    smsJob.Sender.Company = "My Company";
    smsJob.Sender.Name = "Test Person";
    smsJob.Sender.ReplyTo = "+61 4 1234 5678";
    smsJob.IsTwoWay = false;
    smsJob.Recipients = recipientList.ToArray();
    smsJob.Text = "Test SMS message - your favourite animal is
                  %%FavouriteAnimal%%.";

    AuthenticationHeader authHeader = new AuthenticationHeader();
    authHeader.Username = "test";
    authHeader.Password = "test";

    WelWebService service = new WelWebService();
    service.AuthenticationHeaderValue = authHeader;
    int jobid = service.SubmitJob(smsJob);
}
```



```
public void TestTwoWaySMS()
{
    List<SmsRecipient> recipientList = new List<SmsRecipient>();

    SmsRecipient recipient = new SmsRecipient();
    recipient.Title = "Ms";
    recipient.FirstName = "Annette";
    recipient.LastName = "Citizen";
    recipient.Reference = "Of the world";
    recipient.Destination = "+61 4 9876 5432";

    List<MergeField> fieldList = new List<MergeField>();
    MergeField mergeField = new MergeField();
    mergeField.Key = "FavouriteAnimal";
    mergeField.Value = "Cats";
    fieldList.Add(mergeField);

    recipient.MergeFields = fieldList.ToArray();
    recipientList.Add(recipient);

    SmsJob smsJob = new SmsJob();
    smsJob.Name = "2 Way SMS Test";
    smsJob.Sender = new Sender();
    smsJob.Sender.Company = "My Company";
    smsJob.Sender.Name = "Test Person";
    smsJob.Sender.ReplyTo = "smsreplies@mycompany.com";
    smsJob.IsTwoWay = true;
    smsJob.BatchReplies = false;
    smsJob.ExpiryHours = 24;
    smsJob.Recipients = recipientList.ToArray();
    smsJob.Text = "Test SMS message - your favourite animal is
                  %%FavouriteAnimal%%. Reply to this to get 25% off
                  your next visit to Pet Care.";

    AuthenticationHeader authHeader = new AuthenticationHeader();
    authHeader.Username = "test";
    authHeader.Password = "test";

    WelWebService service = new WelWebService();
    service.AuthenticationHeaderValue = authHeader;
    int jobid = service.SubmitJob(smsJob);
}
```

OZMEDIA

```
public void TestTTS()
{
    List<TtsRecipient> recipientList = new List<TtsRecipient>();

    TtsRecipient recipient = new TtsRecipient();
    recipient.Title = "Ms";
    recipient.FirstName = "Annette";
    recipient.LastName = "Citizen";
    recipient.Reference = "Of the world";
    recipient.Destination = "+61 4 9876 5432";

    List<MergeField> fieldList = new List<MergeField>();
    MergeField mergeField = new MergeField();
    mergeField.Key = "FavouriteAnimal";
    mergeField.Value = "Cats";
    fieldList.Add(mergeField);

    recipient.MergeFields = fieldList.ToArray();

    recipientList.Add(recipient);

    TtsJob ttsJob = new TtsJob();
    ttsJob.Name = "Test TTS";
    ttsJob.Sender = new Sender();
    ttsJob.Sender.Company = "My Company";
    ttsJob.Sender.Name = "Test Person";
    ttsJob.Sender.ReplyTo = "+61 4 1234 5678";
    ttsJob.IsTwoWay = false;

    ttsJob.Recipients = recipientList.ToArray();
    ttsJob.Text = "Test TTS message - your favourite animal is
        %%FavouriteAnimal%%.";

    AuthenticationHeader authHeader = new AuthenticationHeader();
    authHeader.Username = "test";
    authHeader.Password = "test";

    WelWebService service = new WelWebService();
    service.AuthenticationHeaderValue = authHeader;
    int jobid = service.SubmitJob(ttsJob);
}
```

OZMEDIA

```
public void TestTTSTwoWay()
{
    List<TtsRecipient> recipientList = new List<TtsRecipient>();

    TtsRecipient recipient = new TtsRecipient();
    recipient.Title = "Ms";
    recipient.FirstName = "Annette";
    recipient.LastName = "Citizen";
    recipient.Reference = "Of the world";
    recipient.Destination = "02 9123 4567";

    List<MergeField> fieldList = new List<MergeField>();
    MergeField mergeField = new MergeField();
    mergeField.Key = "FavouriteAnimal";
    mergeField.Value = "Cats";
    fieldList.Add(mergeField);

    recipient.MergeFields = fieldList.ToArray();

    recipientList.Add(recipient);

    TtsJob ttsJob = new TtsJob();
    ttsJob.Name = "Test Two Way TTS";
    ttsJob.Sender = new Sender();
    ttsJob.Sender.Company = "My Company";
    ttsJob.Sender.Name = "Joe Blow";
    ttsJob.Sender.ReplyTo = "02 9123 4567";

    ttsJob.IsTwoWay = true;
    List<TransferKey> transKeys = new List<TransferKey>();
    TransferKey transKey = new TransferKey();
    transKey.Keypress = 1;
    transKey.MaxSimultaneousCalls = 15;
    transKey.TransferNumber = "03 9123 4567";
    transKeys.Add(transKey);

    ttsJob.TransferKeys = transKeys.ToArray();
    ttsJob.Recipients = recipientList.ToArray();
    ttsJob.Text = "Test TTS message - your favourite animal is
                  %%FavouriteAnimal%%. To speak to a vet, please
                  press 1 now.";

    AuthenticationHeader authHeader = new AuthenticationHeader();
    authHeader.Username = "test";
    authHeader.Password = "test";

    WelWebService service = new WelWebService();
    service.AuthenticationHeaderValue = authHeader;
    int jobid = service.SubmitJob(ttsJob);
}
```

OZMEDIA

```
public void TestVoice()
{
    List<VoiceRecipient> recipientList = new List<VoiceRecipient>();

    VoiceRecipient recipient = new VoiceRecipient();
    recipient.Title = "Ms";
    recipient.FirstName = "Annette";
    recipient.LastName = "Citizen";
    recipient.Reference = "Of the world";
    recipient.Destination = "02 9123 4567";

    recipientList.Add(recipient);

    File myFile = new File();
    myFile.Name = "test.wav";
    myFile.Priority = 1;
    System.IO.FileStream file = new
        System.IO.FileStream("files/test.wav",
            System.IO.FileMode.Open, System.IO.FileAccess.Read);
    System.IO.BinaryReader br = new System.IO.BinaryReader(file);
    byte[] buffer = new byte[(int)file.Length];
    br.Read(buffer, 0, (int)file.Length);
    br.Close();

    myFile.Content = buffer;

    VoiceJob voiceJob = new VoiceJob();
    voiceJob.Name = "WAV File Test";
    voiceJob.Sender = new Sender();
    voiceJob.Sender.Company = "My Company";
    voiceJob.Sender.Name = "My Name";

    voiceJob.Recipients = recipientList.ToArray();
    voiceJob.Recording = myFile;

    AuthenticationHeader authHeader = new AuthenticationHeader();
    authHeader.Username = "test";
    authHeader.Password = "test";

    WelWebService service = new WelWebService();
    service.AuthenticationHeaderValue = authHeader;
    int jobid = service.SubmitJob(voiceJob);
}
```



```
public void GetReport()  
{  
    AuthenticationHeader authHeader = new AuthenticationHeader();  
    authHeader.Username = "test";  
    authHeader.Password = "test";  
  
    WelWebService service = new WelWebService();  
    service.AuthenticationHeaderValue = authHeader;  
    StatusReport statusReport = service.RetrieveReport(1234);  
}
```

```
public void GetDetailedReport()  
{  
    AuthenticationHeader authHeader = new AuthenticationHeader();  
    authHeader.Username = "test";  
    authHeader.Password = "test";  
  
    WelWebService service = new WelWebService();  
    service.AuthenticationHeaderValue = authHeader;  
    DetailedReport statusReport =  
        service.RetrieveDetailedReport(1234);  
}
```



Java Axis Examples

Connecting to the Web Service using Java and Axis 1.4

This document assumes that java and axis are setup appropriately, and the user has some basic familiarity with them.

1. Create the Web Service interface classes:

To create the Webservice class files and the service Stub run:

```
java org.apache.axis.wsdl.WSDL2Java (WSDL-file-URL)
```

as per the axis documentation, then import the resultant classes into your java project.

2. Update Webservice stubs to provide the required Authentication Header.

In the two stub files:

```
Wel_x0020_Web_x0020_ServiceSoapStub  
Wel_x0020_Web_x0020_ServiceSoap12Stub
```

Insert the below code lines after the statement:

```
setRequestHeaders(_call);
```

in the **submitJob**, **retrieveReport** and **retrieveDetailedReport** methods

```
org.apache.axis.message.SOAPHeaderElement auth = new  
SOAPHeaderElement("http://www.welcorp.com/webservice/", "AuthenticationHeader")  
;  
auth.addChildElement("Username", "").addTextNode("test");  
auth.addChildElement("Password", "").addTextNode("test");  
_call.addHeader(auth);
```



3. Main Sample Code

```
/*
 * Main.java
 *
 * Created on 24 November 2006, 20:51
 *
 */

package weltest;

import com.welcorp.www.webservice.*;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Calendar;
import java.util.LinkedList;
import org.apache.axis.types.UnsignedByte;

public class Main {

    public Main() {
    }

    public static void main(String[] args) {
        try {
            TestFax();
            //TestEmail();
            //TestSMS();
            //TestTwoWaySMS();
            //testTTS();
            //TestTTSTwoWay();
            //TestVoice();
            //GetReport();
            //GetDetailsReport();
        } catch (Exception e){
            System.err.println(e.toString());
        }
    }
}
```

OZMEDIA

```
static public void TestFax() throws IOException{
    // Add Fax Recipients
    LinkedList recipientList = new LinkedList();
    FaxRecipient recipient = new FaxRecipient();
    recipient.setTitle("Ms");
    recipient.setFirstName("Annette");
    recipient.setLastName("Citizen");
    recipient.setReference("Of the world");
    recipient.setDestination("02 9123 4567");
    recipientList.add(recipient);
    LinkedList fileList = new LinkedList();

    // Add Files to Fax
    File myFile = new File();
    myFile.setName("test.doc");
    myFile.setPriority(1);

    java.io.File inFile = new java.io.File("c:/test.doc");
    InputStream is = new FileInputStream(inFile);
    long iFlength = inFile.length();
    byte[] buffer = new byte[(int)iFlength];
    int offset = 0;
    int numRead = 0;
    while (offset < buffer.length && (numRead=is.read(buffer, offset,
buffer.length-offset)) >= 0) offset += numRead;
    is.close();

    myFile.setContent(buffer);
    fileList.add(myFile);

    //Set Sender Details
    Sender sender = new Sender();
    sender.setCompany("My Company");
    sender.setName("My Name");
    sender.setReplyTo("02 9876 5432");

    //Setup FaxJob and add sender, recipients and files
    FaxJob faxJob = new FaxJob();
    faxJob.setName("Test Fax");
    faxJob.setSender(sender);
    faxJob.setRecipients((FaxRecipient[])recipientList.toArray(new
FaxRecipient[1]));
    faxJob.setFiles((File[])fileList.toArray(new File[1]));

    Calendar nextWeek = Calendar.getInstance();
    nextWeek.add(Calendar.DAY_OF_WEEK, 7);
    faxJob.setScheduled(nextWeek);

    try {
        int irect = SendJob(faxJob);
        System.out.println(irect);
    } catch (Exception e) {
        System.err.println(e.toString());
    }
}
```

OZMEDIA

```
static public void TestEmail() throws IOException{
    // Add Recipients
    LinkedList recipientList = new LinkedList();
    EmailRecipient recipient = new EmailRecipient();
    recipient.setTitle("Ms");
    recipient.setFirstName("Annette");
    recipient.setLastName("Citizen");
    recipient.setReference("Of the world");
    recipient.setDestination("a.citizen@theworld.com ");

    //Create and add merge fields for recipient, then record recipient
    LinkedList fieldList = new LinkedList();
    MergeField mergeField = new MergeField();
    mergeField.setKey("FavouriteAnimal");
    mergeField.setValue("Cats");
    fieldList.add(mergeField);
    recipient.setMergeFields((MergeField[])fieldList.toArray(new
MergeField[1]));
    recipientList.add(recipient);

    //Add attachment files
    LinkedList fileList = new LinkedList();
    File myFile = new File();
    myFile.setName("test.jpg");
    myFile.setPriority(1);

    java.io.File inFile = new java.io.File("c:/test.jpg");
    InputStream is = new FileInputStream(inFile);
    long iFlength = inFile.length();
    byte[] buffer = new byte[(int)iFlength];
    int offset = 0;
    int numRead = 0;
    while (offset < buffer.length && (numRead=is.read(buffer, offset,
buffer.length-offset)) >= 0) offset += numRead;
    is.close();

    myFile.setContent(buffer);
    fileList.add(myFile);

    //Set Sender Details
    Sender sender = new Sender();
    sender.setName("Test Person");
    sender.setCompany("My Company");
    sender.setReplyTo("test@test.com");

    //Setup EmailJob and add sender, recipients and files
    EmailJob emailJob = new EmailJob();
    emailJob.setName("Test Email");
    emailJob.setSender(sender);
    emailJob.setSubject("My Test Email");
    emailJob.setRecipients((EmailRecipient[])recipientList.toArray(new
EmailRecipient[1]));
    emailJob.setText("Hi %%FirstName%%, your favourite animal is
%%FavouriteAnimal%%.");
    emailJob.setFiles((File[])fileList.toArray(new File[1]));

    Calendar nextWeek = Calendar.getInstance();
    nextWeek.add(Calendar.DAY_OF_WEEK, 7);
    emailJob.setScheduled(nextWeek);

    try {
        int irect = SendJob(emailJob);
        System.out.println(irect);
    } catch (Exception e) {
        System.err.println(e.toString());
    }
}
```

OZMEDIA

```
static public void TestSMS(){
    // Add Recipients
    LinkedList recipientList = new LinkedList();
    SmsRecipient recipient = new SmsRecipient();
    recipient.setTitle("Ms");
    recipient.setFirstName("Annette");
    recipient.setLastName("Citizen");
    recipient.setReference("Of the world");
    recipient.setDestination("+61 4 9876 5432");

    //Create and add merge fields for recipient, then record recipient
    LinkedList fieldList = new LinkedList();
    MergeField mergeField = new MergeField();
    mergeField.setKey("FavouriteAnimal");
    mergeField.setValue("Cats");
    fieldList.add(mergeField);
    recipient.setMergeFields((MergeField[])fieldList.toArray(new
MergeField[1]));
    recipientList.add(recipient);

    //Set Sender Details
    Sender sender = new Sender();
    sender.setName("Test Person");
    sender.setCompany("My Company");
    sender.setReplyTo("+61 4 1234 5678");

    //Setup SmsJob and add sender, recipients and files
    SmsJob smsJob = new SmsJob();
    smsJob.setName("SMS Test");
    smsJob.setSender(sender);
    smsJob.setIsTwoWay(false);
    smsJob.setRecipients((SmsRecipient[])recipientList.toArray(new
SmsRecipient[1]));
    smsJob.setText("Test SMS message - your favourite animal is
%%FavouriteAnimal%%.");

    Calendar nextWeek = Calendar.getInstance();
    nextWeek.add(Calendar.DAY_OF_WEEK, 7);
    smsJob.setScheduled(nextWeek);

    try {
        int iret = SendJob(smsJob);
        System.out.println(iret);
    } catch (Exception e) {
        System.err.println(e.toString());
    }
}
```

OZMEDIA

```
static public void TestTwoWaySMS(){
    // Add Recipients
    LinkedList recipientList = new LinkedList();
    SmsRecipient recipient = new SmsRecipient();
    recipient.setTitle("Ms");
    recipient.setFirstName("Annette");
    recipient.setLastName("Citizen");
    recipient.setReference("Of the world");
    recipient.setDestination("+61 4 9876 5432");

    //Create and add merge fields for recipient, then record recipient
    LinkedList fieldList = new LinkedList();
    MergeField mergeField = new MergeField();
    mergeField.setKey("FavouriteAnimal");
    mergeField.setValue("Cats");
    fieldList.add(mergeField);
    recipient.setMergeFields((MergeField[])fieldList.toArray(new
MergeField[1]));
    recipientList.add(recipient);

    //Set Sender Details
    Sender sender = new Sender();
    sender.setName("Test Person");
    sender.setCompany("My Company");
    sender.setReplyTo("smsreplies@mycompany.com");

    //Setup SmsJob and add sender, recipients and files
    SmsJob smsJob = new SmsJob();
    smsJob.setName("SMS Test");
    smsJob.setSender(sender);
    smsJob.setIsTwoWay(true);
    smsJob.setExpiryHours(new UnsignedByte(24));
    smsJob.setRecipients((SmsRecipient[])recipientList.toArray(new
SmsRecipient[1]));
    smsJob.setText("Test SMS message - your favourite animal is
%%FavouriteAnimal%%." +
        "Reply to this to get 25% off your next visit to Pet Care.");

    Calendar nextWeek = Calendar.getInstance();
    nextWeek.add(Calendar.DAY_OF_WEEK, 7);
    smsJob.setScheduled(nextWeek);

    try {
        int iret = SendJob(smsJob);
        System.out.println(iret);
    } catch (Exception e) {
        System.err.println(e.toString());
    }
}
```

OZMEDIA

```
static public void testTTS(){
    // Add Recipients
    LinkedList recipientList = new LinkedList();
    TtsRecipient recipient = new TtsRecipient();
    recipient.setTitle("Ms");
    recipient.setFirstName("Annette");
    recipient.setLastName("Citizen");
    recipient.setReference("Of the world");
    recipient.setDestination("+61 4 9876 5432");

    //Create and add merge fields for recipient, then record recipient
    LinkedList fieldList = new LinkedList();
    MergeField mergeField = new MergeField();
    mergeField.setKey("FavouriteAnimal");
    mergeField.setValue("Cats");
    fieldList.add(mergeField);
    recipient.setMergeFields((MergeField[])fieldList.toArray(new
MergeField[1]));
    recipientList.add(recipient);

    //Set Sender Details
    Sender sender = new Sender();
    sender.setName("Test Person");
    sender.setCompany("My Company");
    sender.setReplyTo("+61 4 1234 5678");

    //Setup TTSJob and add sender, recipients and files
    TtsJob ttsJob = new TtsJob();
    ttsJob.setName("TTS Test");
    ttsJob.setSender(sender);
    ttsJob.setIsTwoWay(false);
    ttsJob.setRecipients((TtsRecipient[])recipientList.toArray(new
TtsRecipient[1]));
    ttsJob.setText("Test TTS message - your favourite animal is
%%FavouriteAnimal%%.");
    ttsJob.setPreferredDestinationType(PreferredDestinationType.UseDefault);

    Calendar nextWeek = Calendar.getInstance();
    nextWeek.add(Calendar.DAY_OF_WEEK, 7);
    ttsJob.setScheduled(nextWeek);

    try {
        int iret = SendJob(ttsJob);
        System.out.println(iret);
    } catch (Exception e) {
        System.err.println(e.toString());
    }
}
```

OZMEDIA

```
static public void TestTTSTwoWay(){
    // Add Recipients
    LinkedList recipientList = new LinkedList();
    TtsRecipient recipient = new TtsRecipient();
    recipient.setTitle("Ms");
    recipient.setFirstName("Annette");
    recipient.setLastName("Citizen");
    recipient.setReference("Of the world");
    recipient.setDestination("02 9123 4567");

    //Create and add merge fields for recipient, then record recipient
    LinkedList fieldList = new LinkedList();
    MergeField mergeField = new MergeField();
    mergeField.setKey("FavouriteAnimal");
    mergeField.setValue("Cats");
    fieldList.add(mergeField);
    recipient.setMergeFields((MergeField[])fieldList.toArray(new
MergeField[1]));
    recipientList.add(recipient);

    //Set Sender Details
    Sender sender = new Sender();
    sender.setName("Test Person");
    sender.setCompany("My Company");
    sender.setReplyTo("02 9876 5432");

    //Setup TransferKeys
    LinkedList transKeys = new LinkedList();
    TransferKey transKey = new TransferKey();
    transKey.setKeypress(new UnsignedByte(1));
    transKey.setMaxSimultaneousCalls(15);
    transKey.setTransferNumber("03 9123 8765");
    transKeys.add(transKey);

    //Setup TTSJob and add sender, recipients and files
    TtsJob ttsJob = new TtsJob();
    ttsJob.setName("Test Two Wat TTS");
    ttsJob.setSender(sender);
    ttsJob.setIsTwoWay(true);
    ttsJob.setTransferKeys((TransferKey[])transKeys.toArray(new
TransferKey[1]));
    ttsJob.setRecipients((TtsRecipient[])recipientList.toArray(new
TtsRecipient[1]));
    ttsJob.setText("Test TTS message - your favourite animal is
%%FavouriteAnimal%%. " +
        "your favourite animal is %%FavouriteAnimal%%. To speak to a vet,
please press 1 now.");
    ttsJob.setPreferredDestinationType(PreferredDestinationType.UseDefault);

    Calendar nextWeek = Calendar.getInstance();
    nextWeek.add(Calendar.DAY_OF_WEEK, 7);
    ttsJob.setScheduled(nextWeek);

    try {
        int iret = SendJob(ttsJob);
        System.out.println(iret);
    } catch (Exception e) {
        System.err.println(e.toString());
    }
}
```

OZMEDIA

```
static public void TestVoice() throws Exception{
    // Add Recipients
    LinkedList recipientList = new LinkedList();
    VoiceRecipient recipient = new VoiceRecipient();
    recipient.setTitle("Ms");
    recipient.setFirstName("Annette");
    recipient.setLastName("Citizen");
    recipient.setReference("Of the world");
    recipient.setDestination("02 9123 4567");
    recipientList.add(recipient);

    // Add Voice File
    File myFile = new File();
    myFile.setName("test.wav");
    myFile.setPriority(1);

    // Add Files to Fax
    java.io.File inFile = new java.io.File("c:/test.wav");
    InputStream is = new FileInputStream(inFile);
    long iFlength = inFile.length();
    byte[] buffer = new byte[(int)iFlength];
    int offset = 0;
    int numRead = 0;
    while (offset < buffer.length && (numRead=is.read(buffer, offset,
buffer.length-offset)) >= 0) offset += numRead;
    is.close();

    myFile.setContent(buffer);

    //Set Sender Details
    Sender sender = new Sender();
    sender.setCompany("My Company");
    sender.setName("My Name");

    //Setup VoiceJob and add sender, recipients and files
    VoiceJob voiceJob = new VoiceJob();
    voiceJob.setName("WAV File Test");
    voiceJob.setSender(sender);
    voiceJob.setRecipients((VoiceRecipient[])recipientList.toArray(new
VoiceRecipient[1]));
    voiceJob.setRecording(myFile);
    voiceJob.setPreferredDestinationType(PreferredDestinationType.UseDefault);

    Calendar nextWeek = Calendar.getInstance();
    nextWeek.add(Calendar.DAY_OF_WEEK, 7);
    voiceJob.setScheduled(nextWeek);

    try {
        int ired = SendJob(voiceJob);
        System.out.println(ired);
    } catch (Exception e) {
        System.err.println(e.toString());
    }
}
```



```
static public void GetReport() throws Exception{

    // Make a service
    Wel_x0020_Web_x0020_Service service = new
Wel_x0020_Web_x0020_ServiceLocator();
    // Now use the service to get a stub which implements the SDI.
    Wel_x0020_Web_x0020_ServiceSoap WelWebService =
service.getWel_x0020_Web_x0020_ServiceSoap();

    StatusReport report = WelWebService.retrieveReport(1234);

    System.out.println(report.getJobStatus());
}

static public void GetDetailsReport() throws Exception{
    // Make a service
    Wel_x0020_Web_x0020_Service service = new
Wel_x0020_Web_x0020_ServiceLocator();
    // Now use the service to get a stub which implements the SDI.
    Wel_x0020_Web_x0020_ServiceSoap WelWebService =
service.getWel_x0020_Web_x0020_ServiceSoap();

    StatusReport report = WelWebService.retrieveDetailedReport(1234);

    System.out.println(report.getJobStatus());
}

static public int SendJob(Job oJob) throws Exception{
    // Make a service
    Wel_x0020_Web_x0020_Service service = new
Wel_x0020_Web_x0020_ServiceLocator();
    // Now use the service to get a stub which implements the SDI.
    Wel_x0020_Web_x0020_ServiceSoap WelWebService =
service.getWel_x0020_Web_x0020_ServiceSoap();

    int iRes = WelWebService.submitJob(oJob);

    return iRes;
}
}
```



Appendix

Valid number formats

Valid phone, fax or mobile number construction:-

Local number with area code	e.g. 02 9123 4567
International number with “+”	e.g. +44 2 1234 5678
International number with “0011”	e.g. 0011 44 2 1234 5678

Other number constructions are invalid.

Broadcast Status Codes

List Created	User has created their job, but has not yet submitted it.
Pending	For SMS only:- job is in the pending queue.
Retry	For SMS only:- aggregator cannot be contacted, job has been re-queued.
Queued By WS	For Web Service submissions only:- job has been queued and awaiting processing.
Test Sent	For Fax and TTS only:- user has requested a test send and is awaiting results.
Scheduled	Job is scheduled for later date and time.
Broadcast Submitted	Job has been submitted and is currently in progress.
Complete	Job is complete.
Closed	Job has been closed (either completed or cancelled).
Cancelled	Scheduled job has been cancelled by user.



Send Codes

SENT Message was successfully sent to the recipient.

Fax Specific Error Codes

BUSY Line gave busy signal with all retries. The receiving fax machine was busy.

NOAN Phone keeps ringing. Nobody answers. If you are sure that the number is correct, please contact the recipient. There may be a problem with their fax machine (e.g. out of paper).

INVD The number is not a valid number. Please check the number and dial again.

NOLN Problem with phone line. Please check line.

BARR Wrong number (number changed, call barred).

NSUP Service or Function not supported by network or user.

COLL At the moment we started trying to send a fax a call came in.

ERR Procedural or unknown error.

FAIL Document conversion failed. Please resubmit your document.

NOFX No fax machine detected. Please check the number and try again.

PAGC No confirmation received after last page transmitted. The fax may have been received by the recipient and the machine failed to confirm.

BADC Bad connection or fax modem error at receiving end.

Text-To-Speech Specific Error Codes

BSIN Line busy or incorrect number.

MGNC Receipt of message not confirmed.

SMS Specific Error Codes

DERR Aggregator error.

UERR User validation error.

TRAN Transaction limit reached.

PERR Invalid password.

SVRE SMS server error - please retry later.

SNDE Error in sender number.

RECE Error in receiver number.

MSGE No SMS message.

LENE SMS message too long.

NUME Sending number invalid.

OUTE SMS outgoing update error.

Notes:

For SMS: Status SENT indicates that the message has been successfully submitted to the network. In some cases the message may fail after if the number does not exist or for some other network related issue. In these cases, the status will be updated accordingly up to four hours after submission.

For Email: All emails are submitted and the system must wait to either receive bounced back emails, or to receive notification that an email has been read. The system cannot ascertain if all emails are read – if a recipient uses an HTML email package and does not block outgoing code then the “Read” status will be received. However, this is a minority of recipients. After 12 hours the job is closed and all emails that have not failed are updated to “Sent”.