Pitney Bowes

P/I OfficeMail™

Job Submission API

Version: 3.6

Release Date: June 2012

We will continue to maintain this document and we welcome any clarifications or additional information regarding its content. Address comments concerning the content of this publication to:

Pitney Bowes
2nd Floor
6 Hercules Way
Leavesden
Watford
WD25 7GS
UK

Table of Contents

About This Guide

Overview

This documentation describes the required features of the Job Submission API to perform a successful job submission "Upload". Any Datasets described, are those involved in the job submission or in any prerequisites to file upload.

Software

- C# programming language
- MS Visual Studio
- P/I OfficeMail

Operating Systems:

- Microsoft Windows Server 2003
- Microsoft Windows Server 2008
- Microsoft Windows XP
- Microsoft Windows Vista
- Microsoft Windows 7

NOTE: The listed items are not sytem requirements, please refer to the Installation Guide.

Conventions used in this guide

The following is a list of conventions used in this guide. We follow the conventions to help you recognize elements, processes and names that occur frequently in this guide quickly and easily.

Sample Formatting	Element
File menu **Menu bar heading|Menu item|Sub menu item**	Menu options and paths.
`C:\USER\CONFIG\GENERIC.CFG`	Paths, Filenames and File extensions.
Logon dialog	Screen names.
Type your name in the **User Name** field. Click **OK**.	Fields and buttons in a window.
<Enter> **<Ctrl>+<Z>**	Keystrokes.
`a:vipinst c:`	Command line instructions.
`[OBJECT] OBJECT REGION SCANO`	Text in scripts.
NOTE: This is a note.	Notes are asides to the main text containing useful information.
 a warning.	Warnings contain essential information which, if not heeded, may have serious consequences.
Refer to the *Getting Started* section.	The blue text is a hyperlink. If you are reading this document in Adobe Acrobat Reader, clicking the hyperlink text will take you to the relevant section of the guide.

Setup and Configuration

Job Submission API Location

There are two web services which make up the OfficeMail Job Submission API, they are:

UserService	A Web Service comprised of user related operations
JobService	A Web Service comprised of Job Submission related operations

Both of the above services are accessible from the local OfficeMail web server at:

http://xxx/piofficemailuser/piofficemailuserwebservices/jobservice.asmx

http://xxx/piofficemailuser/piofficemailuserwebservices/userservice.asmx

Where

Xxx	the server/IP of the piofficemailuser site

Service References

The above service can be added to a Visual Studio project through the "Add Web Reference" context menu within the Solution Explorer. Once added to the project it can be accessed through the newly created web reference name. See the *"Code Examples" on page 19* for more details.

Job Submission API Authorization

In order to access the web services, any call must first obtain a valid logon. Valid user credentials on the P/I OfficeMail System can be used to obtain an authentication. A user with access to the OfficeMail User Site will have sufficient access to the Job Submission API .

We recommend that a dedicated Job Submission API user is created with the correct permissions for job submissions, whose credentials can be used for the Job Submission API authentication process.

Job Submission API Services

User Service

The user web service (`userservice.asmx`) provides access to user settings and features, such as Print Center information, Printer Driver Version as well as user name, email , addresses etc.

Authentication

This method should be called before any calls are made to either the userservice or the jobservice, below. The method requires a valid username and password, andon successful logon returns an authentication ticket .

The authentication ticket must be sent in the service header for subsequent calls to the userservice and the jobservice.

Method Name: Authenticate

Parameter	In/Out	Data Type	Description
UserName	In	String	The user name used for logging on to OfficeMail
Password	In	String	The password used for logging on to OfficeMail
	Return	String	The Authentication ticket returned

GetGuids

This method returns a pair of comma separated GUIDs specific to the logged on user. The returned GUIDs are used to populate the job ticket for job submissions. The GUID is also used for PDF encryption.

Method Name: GetGuids

Parameter	In/Out	Data Type	Description
	Return	String	Two comma separated guids

GetPrintCentreID

This method provides a mapping between delivery addresses and Print Center IDs in the P/I OfficeMail database. An array of integer IDs is returned for a corresponding array of delivery addresses in the same order.

Method Name: GetPrintCentreID

Parameter	In/Out	Data Type	Description
Addresses	In	String Array	An array of strings representing delivery addresses

	Return	Integer Array	An array of Integers representing the Print Center IDs corresponding to the Print Center Names passed in the input string array. The order of the returned IDs follows the input strings order.

GetPrinterDriverVersion

This method provides Printer Driver information for a given P/I OfficeMail PrinterDriver version, as well as information of the latest driver version available and any driver expiry dates if applicable. This method can be used to check the user's current driver version, and advice on updates.

Method Name: GetPrinterDriverVersion

Parameter	In/Out	Data Type	Description
Version	In	String	The user's version of the PrinterDriver, for example, 1.0.0.25
	Return	PrinterDriverVersionDS	A PrinterDriverVersionDS data set containing information on the current driver supplied in the input parameter, as well as the latest driver available and any expiry dates.

The following data set is returned by the GetPrinterDriverVersion method call.

Data Set Name: PrinterDriverVersionDS

Field	Data Type	Description
LatestPrinterDriver	String	The version of the latest printer driver, this will be the same as the input printer driver supplied unless a newer printer driver version is available
ObsoleteDate	DateTime	The date after which, the supplied printer driver becomes obsolete.
ObsoleteWarningDate	DateTime	The date from which the warning will start that the supplied printer driver should updated.

GetUser

This method provides the largest information set about the logged on user. Due to the large amount of data returned, a data set is returned instead of out parameters. The returned data set is of type UserDS.

Method Name: GetUser

Parameter	In/Out	Data Type	Description
	Return	UserDS	A UserDS data set containing information about the current logged on user.

Data Set Name: UserDS

The user data set is made up of various datasets, each providing information on a specific area of the OfficeMail subsystem.

The datasets that make up the UserDS are:

1. User
2. UserStatistics
3. PrintPlex
4. Printcolor
5. PrintPapeSize
6. MailingService
7. MailingEnvelope
8. MailingEnvelopeWindow
9. MailingEnvelopeContent
10. PageReservedArea
11. JobOptionSend
12. JobOptionNotification
13. JobOptionPageStock
14. JobOptionPageStockReservedArea
15. PrePrintedInsert
16. Attachement
17. TempFileAttachment
18. TempFile
19. TempFileLocation
20. SettingsGroup
21. Mode
22. ModeSupportData
23. ScanningOption
24. PrePaySetting

The following data sets are those that might be used in job submission API calls:

DataSet Name: User

Field	Data Type	Description
UserID	**GUID**	The user's unique GUID –not required for submission
UserName	**String**	The user name
Email	**string**	The user's email address – not required for submission
AuthRequiredLevel	**Integer**	The authorization level of the user
DriverPreviewEdit	**Boolean**	True/False can the user edit in the Driver Preview window

DataSet Name: UserStatistics

Field	Data Type	Description
MailPieceCount	Integer	The number of mail pieces processed for the user

DataSet Name: PrintPlex

Field	Data Type	Description
PrintPlexID	integer	The PrintPlex DB ID
PrintPlex	String	The PrintPlex name, for example, Simplex, Duplex
DisplayOrder	integer	The display order
AuthorizationLevel	Integer	The authorization level required for the user

DataSet Name: Printcolor

Field	Data Type	Description
PrintcolorID	integer	The Printcolor DB ID
Printcolor	String	The Printcolor name, for example, color, Black and White
DisplayOrder	integer	The display order
AuthorizationLevel	Integer	The authorization level required for the user

DataSet Name: PrintPaperSize

Field	Data Type	Description
PrintPaperSizeID	integer	The PrintPaperSize DB ID
PrintPaperSize	String	The PrintPaperSize name , for example, A4 , Letter
DisplayOrder	integer	The display order

DataSet Name: MailingService

Field	Data Type	Description
MailingServiceID	integer	The MailingService DB ID
MailingService	String	The MailingService name , for example, First, Second Class
DisplayOrder	integer	The display order
AuthorizationLevel	Integer	The authorization level required for the user

DataSet Name: MailingEnvelope

Field	Data Type	Description
MailingEnvelopeID	integer	The MailingEnvelope DB ID
MailingEnvelope	String	The MailingEnvelope name, for example, C5
DisplayOrder	integer	The display order
AuthorizationLevel	Integer	The authorization level required for the user

DataSet Name: JobOptionSend

Field	Data Type	Description
JobOptionSendID	integer	The JobOptionSend DB ID
JobOptionSend	String	The JobOptionSend name, for example, Today, Tomorrow
DisplayOrder	integer	The display order
AuthorizationLevel	Integer	The authorization level required for the user

DataSet Name: JobOptionNotification

Field	Data Type	Description
JobOptionNotificationID	integer	The JobOptionNotification DB ID
JobOptionNotification	String	The JobOptionNotification name, for example, On Error, Never
DisplayOrder	integer	The display order
AuthorizationLevel	Integer	The authorization level required for the user

DataSet Name: JobOptionPageStock

Field	Data Type	Description
JobOptionPageStockID	integer	The JobOptionPageStock DB ID
JobOptionPageStock	String	The JobOptionPageStock name, for example, Letter Head, Plain Paper
IsPage1Stock	Boolean	Is the page stock for the first page?
IsPagenStock	Boolean	Is the page stock for subsequent pages?
IsSimplexOnly	Boolean	Is the page stock for Simplex type only?
OverlayFileName	String	The filename of the Overlay to be used with the mail piece
OverlayFileLastUpdatedDate	DateTime	The date the overlay file was last updated
OverlaySideAssignmentID	Integer	

AuthorizationLevel	Integer	The authorization level required for the user
IsPhysicalStock	Boolean	Is the page stock a physical stock or electronic

DataSet Name: Mode

Field	Data Type	Description
ModeID	integer	The Mode DB ID
ModeName	String	The Mode name, for example, Mode1
IsDetected	Boolean	Has the mode been detected

DataSet Name: ModeSupportData

Field	Data Type	Description
ID	Integer	ModeSupportData DB ID
ModeSupportDataName	integer	The ModeSupportData Name, for example, NONE
ModeSupportDataValue	String	The ModeSupportData Value, for example, No Scanning
GroupName	Boolean	The ModeSupportData Group Name, for example, ScanningOption
ModeID	Integer	The Mode ID, for example, 1

Job Service

The Job web service contains operations related to job submission and retrieval of attachments, Overlays and temp files from the P/I OfficeMail system.

The only operation required for submitting jobs is the Upload operation, which is described below in detail.

Upload

This method allows the submission of PDF files to the P/I OfficeMail system for processing. The method requires a base64 string representation of the file be supplied with the input parameters, as well as a Job Ticket data structure populated with the correct values.

The tables below will give example values for each input parameter and data structure field, which will ensure the success of simple job submission of a mail piece without attachments, overlays or Mail Merge.

Method Name: Upload

Parameter	In/Out	Data Type	Description
JobName	In	String	The Job name, for example, "mytestjob.pdf"
FileBase64	In	String	The base64 representation of the PDF file to be uploaded
JobTicket	In	JobTicketDS	The job ticket dataset populated with the relevant data, see below for more details
MailPrice	In	MailPriceDS	The mail price dataset populated with the relevant data fileds

This method has no return value. Exception is thrown on failure. If successful the mail piece count in the UserStatistics dataset should increase.

DataSet Name: MailPrice

Field	Data Type	Description
DiscountAmount	Decimal	For Pre-Pay user: get discount amount from the UserDS MailingPrice dataset For Non Pre-Pay user: leave unset
DiscountCode	String	For Pre-Pay user: get discount Code from the UserDS MailingPrice dataset For Non Pre-Pay user: set to blank
DiscountPercentage	Decimal	For Pre-Pay user: get discount Percentage from the UserDS MailingPrice dataset For Non Pre-Pay user: leave unset
Price	Decimal	For Pre-Pay user: get Price from the UserDS MailingPrice dataset For Non Pre-Pay user: leave unset
PrintCentreID	Integer	For Pre-Pay user: get PrintCentreID from the UserDS MailingPrice dataset For Non Pre-Pay user: set to the ID of the desired PrintCenter
SenderCountryISO 3166Code	String	For Pre-Pay user: get SenderCountryISO3166Code from the UserDS MailingPrice dataset For Non Pre-Pay user: set to Code of the desired Sender Country, for example, "GB"
TariffID	Integer	For Pre-Pay user: get TariffID from the UserDS MailingPrice dataset For Non Pre-Pay user: set 0

TAX	Decimal	For Pre-Pay user: get TAX from the UserDS MailingPrice dataset For Non Pre-Pay user: leave unset
TaxRate	Decimal	For Pre-Pay user: get TaxRate from the UserDS MailingPrice dataset For Non Pre-Pay user: leave unset
Total	Decimal	For Pre-Pay user: get Total from the UserDS MailingPrice dataset For Non Pre-Pay user: leave unset

The user's Pre-Pay status can be determined by examining the UserDS dataset for the PrePaySetting subset which contains the Boolean field "IsPrePayCompany".

DataSet Name: JobTicket

Numeric field value examples given below, assume a corresponding mapping exists in your implementation between IDs and their String representation, the ID values are dependent on the settings in your OfficeMail Admin web site. To illustrate, the following is an example of a Color, Mailing Service Mappings:

color	**ID**
Black and White	1
color	2

Mailing Service	**ID**
First Class	1
Second Class	2

If these options are provided to a User through selection drop-downs, they need to reflect the actual ID values on the server.

Field	**Data Type**	**Description**
AuthorisationLevel	Integer	Set to 9 for highest level (highest required authorisation level for this job)
JobOptionNotificationID	integer	Set to 1 for "On Error"
PrintPlexID	integer	1 for Simplex, 2 for Duplex (user selection)
PrintProductionPlexID	integer	1 for Simplex, 2 for Duplex (required for printing o f this job)
PrintcolorID	Integer	1 for Black and White, 2 for color
PrintPaperSizeID	Integer	1 for A4 paper
MailingServiceID	Integer	1 for First Class, 2 for Second (one of MailingServiceID available from User DataSet)

DestinationAddress	String	The Destination Address, for example, “10 PB Avenue, London , N21 3ED”
Mode1ScanOnlySides	String	Leave blank “”
MailingEnvelopeID	Integer	1 for “C5” (one of MailingEnvelopeID available from the User DataSet)
JobOptionSendID	Integer	1 for “Today”, 2 for “Tomorrow”
JobOptionPage1StockID	Integer	1 for “Plain Paper”
JobOptionPagenStockID	Integer	1 for “Plain Paper”
JobReference	String	A job reference text, for example, “this is my 1st test”
Pages	Integer	The number of pages in the mail document job
Nonce	Integer	This is a unique number that protects against replay attacks, set this to the PrintedPageCount value from the UserStatistics dataset (this part of the UserDS dataset returned by the GetUser API call detailed above).
KeyIndex	GUID	This is the first part of the Guid pair returned by a call to GetGuids() method in the User Service.

Code Examples

User Service Operations

Authenticate

1. Create a web service reference for the user web service at your P/I OfficeMail web server, for example:

   http://myofficemailserver/piofficemailuser/piofficemailuserwebservices/userservice.asmx

2. Add a using statement to the namespace and service reference id you chose in the Add Web Refernce wizard, for example:

   using `OFMAWebServiceTester.userServiceRef`;

3. Create a UserService object and make API calls to the User Web Service as below:

```
try
{
//instantiate a userservice object
userServiceRef.UserService userService=new UserService();
//call authenticate with the username and password, save the
authentication ticket string
string AuthTicket=userService.Authenticate(Login1.UserName.Trim(),
Login1.Password.Trim());
//validate the authenticatin ticket
if (String.IsNullOrEmpty(AuthTicket))
{
    //inform the user of the failed logon
 Login1.FailureText="User/Password is not valid!";

    //if this called in an ASP.NET Login control handler method
 //then indicate that formsauthentication has failed
 e.Authenticated=false;
 }
 else
 {
    //authentication has succeeded, save the the ticket to be used
    //in the service header for subsequent API calls
 Session["Ticket"]=AuthTicket;
 e.Authenticated=true;

    }
}
catch (Exception ex)
{
    //handle any exceptions, SoapException are likey if there is
 //a communication error with the webservice
 }
```

GetGuid

1. Create a web service reference for the user web service at your P/I OfficeMail web server, for example:

   http://myofficemailserver/piofficemailuser/piofficemailuserwebservices/userservice.asmx

2. Add a using statement to the namespace and service reference id you chose in the Add Web Refernce wizard, for example:

   using `OFMAWebServiceTester.userServiceRef`;

3. Create a UserService object and make API calls to the User Web Service as below:

```
try
{
    //instatntiate a userservice object
userServiceRef.UserService usersvc=new
OFMAWebServiceTester.userServiceRef.UserService();
    //populate the service header with the authentication ticket
saved from the Authenticate API call
    userServiceRef.ServiceAuthenticationHeader header=new
OFMAWebServiceTester.userServiceRef.ServiceAuthenticationHeader()
;
    header.AuthenticationTicket=Session["Ticket"].ToString();
     //set the header value
     usersvc.ServiceAuthenticationHeaderValue=header;
     //call the GetGuids method, save the returned guids string
     string guids=usersvc.GetGuids();
     //validate the returned guids
     if (!String.IsNullOrEmpty(guids))
 {
       //do something with the returned guids
 }
 }
 catch (Exception ex)
 {
     //catch exceptions .. SoapExceptions are common if a
communication error is raised
 Label_GetGuids.Text=ex.Message;
 }
```

GetPrintCenterID

1. Create a web service reference for the user web service at your P/I OfficeMail web server, for example:

   http://myofficemailserver/piofficemailuser/piofficemailuserwebservices/userservice.asmx

2. Add a using statement to the namespace and service reference id you chose in the Add Web Refernce wizard, for example:

   using `OFMAWebServiceTester.userServiceRef`;

3. Create a UserService object and make API calls to the User Web Service as below:

```
 try
 {
    //instatntiate a userservice object
```

```
 userServiceRef.UserService usersvc=new
OFMAWebServiceTester.userServiceRef.UserService();
    //populate the service header with the authentication ticket
saved from the Authenticate API call
    userServiceRef.ServiceAuthenticationHeader header=new
OFMAWebServiceTester.userServiceRef.ServiceAuthenticationHeader()
;
    header.AuthenticationTicket=Session["Ticket"].ToString();
    //set the header value
    usersvc.ServiceAuthenticationHeaderValue=header;

    //create the print centers string array
    string[] sPrintCenters={ TextBx_PrintCenter.Text };
    //call the GetPrintCenterID method, passing the print centers
string array
    int[] PrintCenterIDs=usersvc.GetPrintCentreID(sPrintCenters);
    //validate the returned array of IDs
    if (PrintCenterIDs.Length > 0)
    {
    //do something with the returned IDs
    Label_GetPrintCenterID.Text=PrintCenterIDs[0].ToString();
    }
}
catch (Exception ex)
{
    //catch exceptions .. SoapExceptions are common if a
communication error is raised
Label_GetPrintCenterID.Text=ex.Message;
}
```

Get PrinterDriverVersion

1. Create a web service reference for the user web service at your P/I OfficeMail web server, for example:

   http://myofficemailserver/piofficemailuser/piofficemailuserwebservices/userservice.asmx

2. Add a using statement to the namespace and service reference id you chose in the Add Web Refernce wizard, for example:

   using `OFMAWebServiceTester.userServiceRef;`

3. Create a UserService object and make API calls to the User Web Service as below:

```
try
{
    //instatntiate a userservice object
    userServiceRef.UserService usersvc=new
OFMAWebServiceTester.userServiceRef.UserService();
    //populate the service header with the authentication ticket
saved from the Authenticate API call
    userServiceRef.ServiceAuthenticationHeader header=new
OFMAWebServiceTester.userServiceRef.ServiceAuthenticationHeader()
;
    header.AuthenticationTicket=Session["Ticket"].ToString();
    //set the header value
    usersvc.ServiceAuthenticationHeaderValue=header;
```

```
    //call the PrinterDriverVersion method, with the current
driver version as input
    userServiceRef.PrinterDriverVersionDS
pdDS=usersvc.GetPrinterDriverVersion(TextBox_PrinterDriverVersion
.Text);
    //validate the retruned PrinterDriverVersionDS dataset
    if (pdDS.PrinterDriverVersion.Count > 0)
    {
    //use the retruned PrinterDriverVersionDS dataset
    Label_GetPrinterDriverVersion.Text="Latest Driver Version:" +
pdDS.PrinterDriverVersion[0].LatestPrinterDriver;
    Label_GetPrinterDriverVersion.Text += "Your driver will expire
on:" +
pdDS.PrinterDriverVersion[0].ObsoleteWarningDate.ToShortDateStrin
g();
    }
}
catch (Exception ex)
{
    //catch exceptions .. SoapExceptions are common if a
communication error is raised
    Label_GetPrinterDriverVersion.Text=ex.Message;
}
```

Get User

1. Create a web service reference for the user web service at your P/I OfficeMail web server, for example:

   http://myofficemailserver/piofficemailuser/
   piofficemailuserwebservicesuserservice.asmx

2. Add a using statement to the namespace and service reference id you chose in the Add Web Refernce wizard, for example:

   ```
   using OFMAWebServiceTester.userServiceRef;
   ```

3. Create a UserService object and make API calls to the User Web Service as below:

```
try
{
    //instatntiate a userservice object
    userServiceRef.UserService usersvc=new
OFMAWebServiceTester.userServiceRef.UserService();
    //populate the service header with the authentication ticket
saved from the Authenticate API call
    userServiceRef.ServiceAuthenticationHeader header=new
OFMAWebServiceTester.userServiceRef.ServiceAuthenticationHeader()
;
    header.AuthenticationTicket=Session["Ticket"].ToString();
    //set the header value
    usersvc.ServiceAuthenticationHeaderValue=header;

    //call the GetUser method, save the returned UserDS dataset
    userServiceRef.UserDS userDS=usersvc.GetUser();
    //validate the returned UserDS dataset
    if (userDS != null)
{
    //use the returned UserDS dataset
```

```
    Label_GetUser.Text=userDS.User.Rows[0]["Email"].ToString();
    }
 }
 catch (Exception ex)
 {
    //catch exceptions .. SoapExceptions are common if a
communication error is raised
    Label_GetUser.Text=ex.Message;
 }
```

Job Service

Upload

1. Create a web service reference for the job web service at your P/I OfficeMail web server, for example:

   http://myofficemailserver/piofficemailuser/piofficemailuserwebservices/jobservice.asmx

2. Add a using statement to the namespace and service reference id you chose in the Add Web Refernce wizard, for example:

   ```
   using OFMAWebServiceTester.jobServiceRef;
   ```

3. Create a jobService object and make API calls to the Job Web Service as below.
4. Ensure you have a valid PDF file to pass to the upload method .

```
try
{
    //get the userDS as it will be needed to fill the JobTicket and
the MailPrice structs
    userServiceRef.UserDS userDS=GetUserDS();
    if (userDS == null)
{
      Label_Upload.Text="Failed to get User DS!";
      return;
}
    //instantiate a jobeservice object
    jobServiceRef.JobService jobsvc=new
OFMAWebServiceTester.jobServiceRef.JobService();
    //populate the service header with the previously save
authentication ticket
    //in the userservice Authenticate call.
    jobServiceRef.ServiceAuthenticationHeader header=new
OFMAWebServiceTester.jobServiceRef.ServiceAuthenticationHeader();
header.AuthenticationTicket=Session["Ticket"].ToString();
    //set the header value
    jobsvc.ServiceAuthenticationHeaderValue=header;

    //1. create and populate the job ticket
    jobServiceRef.JobTicket jt=GetJobTicket(userDS);

    //2. create and populate the mailprice
    jobServiceRef.MailPrice mp=GetMailPrice(userDS)

    //3. get the job name
    string jobName=TextBox_JobName.Text.Trim();
```

```
        //4. get the base64filename
        string base64file="";

        // convert the pdf input file to base64 string
        FileStream inputPDF=new FileStream(@"c:\api-submission-
test.pdf", FileMode.Open, FileAccess.Read);
        long len=inputPDF.Length;
        byte[] buff=new byte[len];
        //get the bytes read
        int byteread=inputPDF.Read(buff, 0, (int)len);
        inputPDF.Close();

        //converts the bytes read to bas64 string
        base64file=System.Convert.ToBase64String(buff);

        //trap file conversion errors
        if (base64file == "")
        {
        Label_Upload.Text="Failed to get source file to upload or source
file could not be converted to base64!";
        }
        //5. upload the file
        jobsvc.Upload(jobName, base64file, jt, mp);
    }
    catch (Exception ex)
    {
        //catch exceptions .. SoapExceptions are common if a
communication error is raised
        Label_Upload.Text=ex.Message;
    }
```

JobTicket Creation Example

```
    private jobServiceRef.JobTicket GetJobTicket(userServiceRef.UserDS
    userDS)
    {
    jobServiceRef.JobTicket jt=new jobServiceRef.JobTicket();
    try
    {
        jt.AuthorisationLevel=9; // set to heighest
        //set the job notification option from UI
        jt.JobOptionNotificationID=1; // on error
        //or from the UserDS if one exists
        for (int i=0; i < userDS.JobOptionNotification.Count; i++)
        {
          if (userDS.JobOptionNotification[i].JobOptionNotification
    == "On Error")
    {
          jt.JobOptionNotificationID=
    userDS.JobOptionNotification[i].JobOptionNotificationID;
          break;
        }
    }
        //set Plex/Duplex .. this is from UI selection ..
        jt.PrintPlexID=1; // PrintPlexID is 1 for Simplex
```

```
    //set PrintProductionPlexID
    jt.PrintProductionPlexID=1;
    //set the color id
    jt.PrintcolorID=1; // BlackAndWhite=1, color=2
    //set the papersize id
    jt.PrintPaperSizeID=1; // i.e A4
    //set mailing service id
    jt.MailingServiceID=1; //First Class option
    //check user ds mailing service if one exists
    for (int i=0; i < userDS.MailingService.Count; i++)
{
      //get the selection from the UI.. combobox etc
      if (userDS.MailingService[i].MailingService == "Second
Class")
{
        jt.MailingServiceID=
userDS.MailingService[i].MailingServiceID;
        break;
}
}

    //set the destination address
    // string destAddress ="10 Downing Street\r\n London\r\n
W10C15";
    jt.DestinationAddress=destAddress
    //set the Mode1ScanOnlySides
    jt.Mode1ScanOnlySides="";
    //set MailingEnvelopeID to a default
    jt.MailingEnvelopeID=1; // C5
    //or get the MailingEnvelopeID from the UserDS dataset
    for (int i=0; i < userDS.MailingEnvelope.Count; i++)
{
      if (userDS.MailingEnvelope[i].MailingEnvelope == "C5")
      {
        jt.MailingEnvelopeID=
userDS.MailingEnvelope[i].MailingEnvelopeID;
        break;
}
}
    //set JobOptionSendID
    jt.JobOptionSendID=1; // to a default
    //or get the JobOptionSendID from the UserDS dataset
    for (int i=0; i < userDS.JobOptionSend.Count; i++)
    {
    if (userDS.JobOptionSend[i].JobOptionSend == "Today")
    {
    jt.JobOptionSendID=userDS.JobOptionSend[i].JobOptionSendID;
    break;
    }
    }
    //if the JobSendOption was set to calendar, then
JobOptionSendDate must be set with the en-US culture
    if (jt.JobOptionSendID == 4) //calendar
    {
    //set the date .. assuming the calendar was set to NOW
```

```
    System.Globalization.CultureInfo cinfo=new
System.Globalization.CultureInfo("en-US", false);
    jt.JobOptionSendDate=new DateTime(DateTime.Now.Year,
DateTime.Now.Month, DateTime.Now.Day, 12, 00, 0, 0,
cinfo.Calendar);
    }
    //set JobOptionPage1StockID to defaults
    jt.JobOptionPage1StockID=1; //plain paper
    string JobOptionPage1Stock="LetterHead";
    string JobOptionPageNStock="Plain Paper";
    bool bPage1StockSet=false, bPageNStockSet=false;
    //or get the JobOptionPage1StockID and JobOptionPagenStockID
from the UserDS dataset
    for (int i=0; i < userDS.JobOptionPageStock.Count; i++)
    {
      if (userDS.JobOptionPageStock[i].JobOptionPageStock ==
JobOptionPage1Stock)
      {
        if (userDS.JobOptionPageStock[i].IsPage1Stock)
        {
          jt.JobOptionPage1StockID=
userDS.JobOptionPageStock[i].JobOptionPageStockID;
          bPage1StockSet=true;
        }
      }
    if (userDS.JobOptionPageStock[i].JobOptionPageStock ==
JobOptionPageNStock)
    {
    if (userDS.JobOptionPageStock[i].IsPagenStock)
    {
        jt.JobOptionPagenStockID=
userDS.JobOptionPageStock[i].JobOptionPageStockID;
        bPageNStockSet=true;
      }
    }
        if (bPage1StockSet & bPageNStockSet)
          break;
      }
      //set the job refernce
      jt.JobReference="myjobreference";;
      //set the numbet of pages in the document
      jt.Pages=1;
      //this prevents replay attacks by using the PrintedSideCount
for the user as a unique number which gets incremented with every
job
      jt.Nonce=userDS.UserStatistics[0].PrintedSideCount;
      //set the key index, get the guids from the User Service
      string Guids=GetUserGuids();
      string IndexGuid=Guids.Remove(36, 37);
      Guid keyIndexGuid=new Guid(IndexGuid);
      jt.KeyIndex=keyIndexGuid;

    }
    catch (Exception ex)
    {
      Label_Upload.Text=ex.Message;
```

```
    }
    return jt;
}
```

MailPrice Creation Example

```
private jobServiceRef.MailPrice GetMailPrice(userServiceRef.UserDS
userDS)
{
    //create the MailPrice object to return
    jobServiceRef.MailPrice mp=new jobServiceRef.MailPrice();
    try
    {
    //get the user's MailingPrice dataset from the UserDS
    userServiceRef.MailingPrice mailingprice=GetMailingPrice();
    //determine pre-pay status
    bool bIsUserPrePay=userDS.PrePaySetting[0].IsPrePayCompany;
    if (bIsUserPrePay)
    {
      //get the user's MailingPrice discount code if one exists
      mp.DiscountAmount=
mailingprice.MailingPriceItems[0].DiscountAmount;
      if (mailingprice.MailingPriceItems[0].DiscountCode != "")
        mp.DiscountCode=
mailingprice.MailingPriceItems[0].DiscountCode;
      else
        mp.DiscountCode="";
      //get the user's MailingPrice discount percentage
      mp.DiscountPercentage=
mailingprice.MailingPriceItems[0].DiscountPercentage;
      //get the user's MailingPrice Price, or set 0 if none exists
      mp.Price=mailingprice.MailingPriceItems[0].Price.HasValue ?
mailingprice.MailingPriceItems[0].Price : new decimal(0);
      //get the user's MailingPrice PrintCentreID
      mp.PrintCentreID=
mailingprice.MailingPriceItems[0].PrintCentreID;
      //get the user's MailingPrice SenderCountryISO3166Code
      mp.SenderCountryISO3166Code=
mailingprice.MailingPriceItems[0].SenderCountryISO3166Code;
      //get the user's MailingPrice TariffID
      mp.TariffID=mailingprice.MailingPriceItems[0].TariffID;
      //get the user's MailingPrice TAX
      mp.TAX=mailingprice.MailingPriceItems[0].TAX;
      //get the user's MailingPrice TaxRate
      mp.TaxRate=mailingprice.MailingPriceItems[0].TaxRate;
      //get the user's MailingPrice Total
      mp.Total=mailingprice.MailingPriceItems[0].Total;
    }
    else
    {
      //set non pre-pay user MailPrice fields
      mp.DiscountCode="";
      mp.SenderCountryISO3166Code="GB";
      mp.PrintCentreID=1; // based on your Admin Site setting
      mp.TariffID=0;
      }
```

```
        }
        catch (Exception ex)
        {
        Label_Upload.Text=ex.Message;
    }
    return mp;
    }
```

Technical Support

You will find full details of the configuration and operation of this product in the user documentation supplied.

Should you encounter any difficulties that you cannot resolve with aid of the user documentation, you will be able to obtain technical support from your supplier, or from Pitney Bowes.

North and South America

Telephone (all products)

ACD Loop 9-5:30 Eastern (active call directory)	**+1 866 220 1014**
After Hours	**+1 877 677 3375**

Email

Output Management products	sptus@pb.com
ADF Products	adfsupport@pb.com

EMEA & Asia Pacific

Telephone (all products)	**+44 (0)1923 279300**
Email (all products)	softwaresolutionsupport@pb.com

In Europe:

2nd Floor
6 Hercules Way
Leavesden
Watford
WD25 7GS
UK
Telephone: +44 (0)1923 279300
Fax: +44 (0)1923 279301

In the US:

37 Executive Drive
Danbury
Connecticut 06810-4182
USA
Telephone: +1 203 792 1600
+1 877 536 2736
Fax: +1 203 739 3704

901 Yamato Road
Suite 120
Boca Raton
Florida 33431
USA
Telephone: +1 561 241 7229
Fax: +1 561 988 9561

Pitney Bowes