



Faxman
Junior

Version 2

Table of Contents

1. Getting Started	1
1.1. Overview	1-2
1.2. Installation	2
1.3. Faxing Process	2-4
1.4. Faxmodem Overview	4-5
1.5. Using the ActiveX Controls	6
1.5.1. tutorial	6-9
1.5.2. FaxMan Jr. Tutorial	10
1.5.3. FaxFinder	10-12
1.5.4. Sending Faxes	12-14
1.5.5. Creating Fax Files	14-15
1.5.6. Fax Resolution	15-16
1.5.7. Sending Options	16
1.5.8. Receiving Faxes	16-17
1.5.9. Event and Status Objects	17-21
1.5.10. Rings	21-22
1.5.11. Cancelling Faxes	22
1.6. Using the .Net Components	22
1.6.1. Quick Start	22-23
1.6.2. Using the Modems Collection	23-24
1.6.3. Creating Fax Files	24-25
1.6.4. Sending Faxes	25-27
1.6.5. Receiving Faxes	27-28
1.6.6. Viewing Faxes	28-31
1.6.7. Using the Printer Control	31-33

1.6.8. Batch Sending of Faxes	33-35
1.6.9. AutoReceiving of Faxes	35-39
1.6.10. Using the Debug Logging Feature	39-40
1.6.11. FaxMan Jr and Visual Studio 2005	40
2. Printer Drivers	41
2.1. Using the FaxMan Printer Drivers	41
2.2. Using the FaxInstall Utility	41-43
2.3. Using the DeletePrn4 Utility	43-44
3. Appendices	45
3.1. Whats new in V 2.00	45
3.2. Distributing the FaxMan ActiveX Components	45-47
3.3. Contacting DTI	47-49
3.4. Converting FMF Files to TIFF Format	49
3.5. Class 2/2.0 Modem Hangup Codes	49-51
3.6. Modem Configuration Information	51-52
3.7. Coverpages	52-54
3.8. Banner And Coverpage Formatting Characters	54-56
3.9. Visual Basic Source Code for Creating a Cover Page	56-62
3.10. C Source Code for Creating a Cover Page	62-64
3.11. Cover Page Reference	64-68
3.12. Error Codes	68-69
3.13. FaxmanJr License	69-71

1 Getting Started

1.1 Overview



Thanks for buying FaxMan Jr., the quick and easy way to send and receive faxes using standard class 1, 2, 2.0 and 2.1 faxmodems. FaxMan Jr. makes it a snap to send faxes from your Windows application or from virtually any other Windows application using the supplied printer drivers.

FaxMan Jr. Technical Support

FaxMan Jr includes 60 days of free phone, email, fax and web support. The sixty day support period starts with your first support call, after that several plans are available to provide various levels of support. Visit our website at **www.data-tech.com** (<http://www.data-tech.com/>) for complete plan details and to order.

Before contacting DTI for technical support, be sure to thoroughly read and understand the FaxMan Jr. documentation, and check our online knowledge base for more up to date information about FaxMan Jr.

Our website also hosts our Online Update Center, which allows you to obtain updates to the latest FaxMan Jr. release. You'll need to have your serial number handy and have registered your product to be able to use this service. You can also sign up for the FaxMan mailing list on our website. Once signed up, you'll receive automatic e-mail notifications of updates and other important information about FaxMan Jr.

Data Techniques, Inc.
PO Box 606
Burnsville, NC 28714

Email: **support@data-tech.com**

Web: **www.data-tech.com** (<http://www.data-tech.com/>)

Tech Support: 828-682-0161 9am to 5pm EST

Sales: 828-682-4111 9am to 5pm EST

Fax: 828-682-0025

1.2 Installation

The install process will allow you to select the destination directory and also what components will be installed. We recommend installing using the default selections of the FaxMan Jr. system at this time, although you are free to install only those portions you wish.

FaxMan Jr. ships with sample code for:

- Visual Basic 6
- C#
- VB.Net
- Visual C++
- Microsoft Access 2003

Once installed you'll have a set of ActiveX controls for use in Visual Basic, C++, MFC and other 32 bit ActiveX containers as well as a set of fully managed .Net assemblies for use with C#, VB.Net and any other .Net language.

1.3 Faxing Process

1.3.1 Quick Fax Primer

In this section we'll give you a quick overview of the faxing process in general. This will help you to better understand the event notifications that FaxMan Jr. sends to your application.

1.3.2 Phases of the Fax Process

The standard faxing process is documented in an exciting little report from the ITU, in which all of the various aspects of a "standard" fax session are described. In this report, the faxing process is broken down into several phases, labeled A

through E, as follows:

Phase A – In this phase, the fax session is initiated. The specification leaves this section somewhat vague, implying that a fax session can be initiated in a number of ways, but practically speaking a fax session is initiated by dialing a phone number and sending that high-pitched fax tone that we’ve all become familiar with. Phase A is completed once both sides are convinced that they’re talking to another fax machine.

Phase B – Phase B is a negotiation phase, in which both fax devices agree on the specific parameters for the fax session (speed, resolution, etc.) It is also in this phase that both fax devices transmit an identification string (referred to in this documentation as the LocalID or RemoteID). This ID string can be up to 20 characters long.

Phase C – The actual data transmission occurs in Phase C. This is where your fax session will generally spend the bulk of its time.

Phase D – This is another negotiation phase, in which the receiving fax device confirms that it received the message OK, and in which the sending device informs the receiver whether there are more pages to be sent. If there are more pages to be sent, the fax session returns to Phase B. Note that while it is theoretically possible for new fax parameters to be negotiated for each page, in practice this very rarely occurs.

Phase E – This is the call disconnect phase. The fax devices agree to stop communicating and the phone line is then hung up.

FaxMan Jr. sends your application notification messages (through the Status/FaxStatus events) throughout each phase of a normal fax session. The following chart illustrates the general order in which you will receive these status event notification messages, for both sending and receiving:

Receiving Events	Sending Events
Initializing Modem	Initializing Modem
Initializing Modem for Receive	Initializing Faxmodem for sending
Waiting for Ring...	Dialing...

Answering...	Waiting to Connect
Negotiating...	Connected to remote fax machine...
Receiving Fax Page data...	Negotiating...
End of Page...	Sending Fax Page...
Send Complete	Sending Page Data...
Comm Port Closed	Sending Page, 10% Complete
	Sending Page, 20% Complete
	Sending Page, 30% Complete
	Sending Page, 40% Complete
	Sending Page, 50% Complete
	Sending Page, 60% Complete
	Sending Page, 70% Complete
	Sending Page, 80% Complete
	Sending Page, 90% Complete
	Sending Page, 100% Complete
	Send Complete...
	Comm Port Closed

Note that not all of these messages are guaranteed to be received by your application. In general, you can only be assured that the FAXST_COMPLETE event will be received when the fax session is completed.

1.4 Faxmodem Overview

The faxing standard so loosely described above details how two facsimile devices may recognize each other and communicate data between themselves. This standard says nothing, however, about how a programmer might communicate with a faxmodem in order to make it send or receive a fax. Thus a new standard was required to allow an application programmer to work with the newly developing breed of faxmodems.

This new standard was called the Class 1 faxmodem standard, and was an

extension of the standard AT modem command set. Also developed by the ITU, it quickly became commonplace in off the shelf faxmodems. Class 1 became so prevalent, in fact, that it is almost impossible to purchase a modem that doesn't support Class 1 commands for faxing.

But Class 1 faxmodems have a problem – they require a lot of attention from the CPU in order to handle timing-sensitive fax operations. This was a distinct problem for the CPU-weak personal computers then available. Another solution was needed which offloaded much of the processing-intensive aspects of faxing onto the faxmodem hardware.

This new standard was to be known as the Class 2 faxmodem standard, and indeed many faxmodem manufacturers began producing "Class 2 compatible" faxmodems even before the Class 2 standard was finalized. What they hadn't counted on, however, was that the Class 2 standard would be held up for several years due to bickering in the standards committees. As a result, a flood of so-called "Class 2" faxmodems entered the marketplace, each one doing its best to meet a standard that was decidedly unfinished. This resulted in a problem with these Class 2 devices: there was no real programming standard for working with them. Any manufacturer that claims their faxmodems support Class 2 is technically lying, as there is no Class 2 standard. Fortunately for us, though there isn't a Class 2 standard, the industry has evolved a handful of Class 2 implementations that are more or less workable and fairly similar to each other, thus allowing faxmodem application developers to devise workarounds that function with most Class 2 devices.

As a result of this Class 2 debacle, when the Class 2 specification was actually finalized, the name of the new spec had to be changed to distinguish it from the many non-compatible Class 2 devices already on the market. Thus the Class 2.0 specification was born. Functionally quite similar to Class 2 operations, Class 2.0 operations are, however, much more standardized.

FaxMan Jr also now supports Class 2.1 modems which add support for faxing at 33.6K which makes faxing over twice as fast as standard 14.4K faxing.

Thus you now have a world wherein you are likely to run into faxmodems that support one or more of these three programming command languages. In fact, most faxmodems sold today support both Class 1 and Class 2.0, while many newer faxmodems also support Class 2.1

FaxMan Jr. works with all three of these specifications, though it defaults when possible to Class 1 due to the increased control it provides.

1.5 Using the ActiveX Controls

1.5.1 tutorial

Quick Start Tutorial

This quick-start tutorial will show you how to quickly add the FaxMan Jr. fax control to your application and begin sending faxes. It should give you enough information to get started...if you have a problem you can always refer to the more extensive tutorial that follows.

We assume that you're working with Visual Basic. If you're not, you should have very little trouble translating this tutorial to your development environment.

Step 1: Create a new VB project.

Create a new VB project with a form that you can drop the FaxMan Jr. control onto.

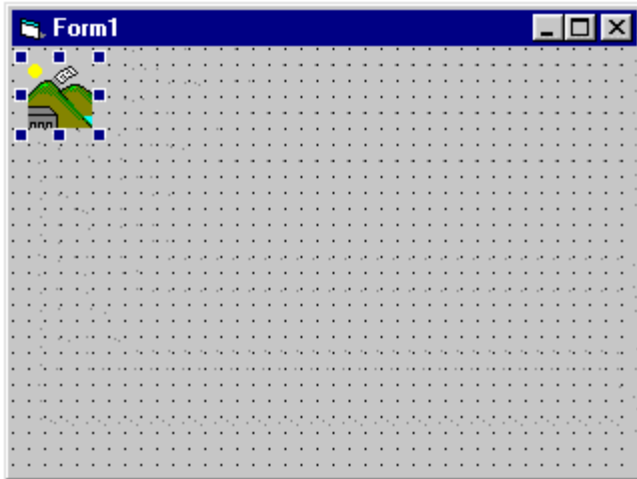
Step 2: Add the FaxMan Jr. controls to your application.

The next thing you need to do, of course, is to add the FaxMan Jr. controls to your control palette so you can then add the FaxMan Jr. control to your form. The illustration below shows the two controls added to your toolbar.



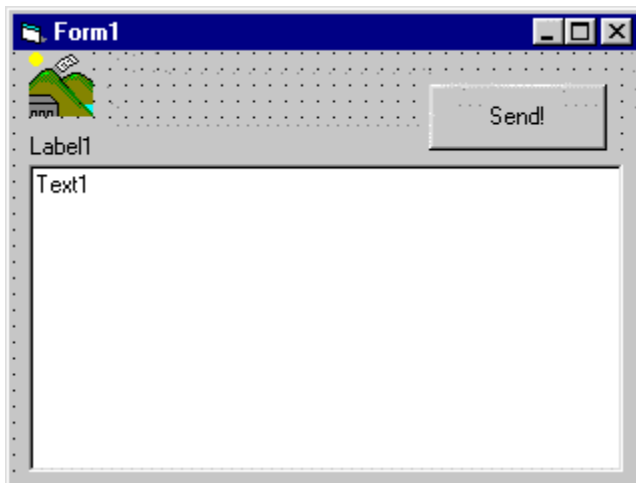
Step 3: Add the FaxMan Jr. control to your form.

Add a new FaxMan Jr. control to your form. It should automatically be named FaxManJr1 – if not, change the name to FaxManJr1.



Step 4: Fill out your form.

Add the rest of the controls you'll need for this quick tutorial: a label control to display returned status values (named Label1), an edit control to display messages returned from FaxMan Jr. (named Text1) and a button to start the whole process (named Command1).



Be sure to setup the text control so that it has a vertical scrollbar and is multi-line.

Step 5: Add sending code to the Send! button.

Double click on the Send button and add the following code:

```
Private Sub Command1_Click()
```

'use the number of your comm port here

```
FaxManJr1.Port = 1
```

'be sure to specify the complete path to

'the file to send

```
FaxManJr1.FaxFiles = "c:\FaxMan\server\FaxMan.fmf"
```

'put your own phone number in here

```
FaxManJr1.FaxNumber = "555-5555"
```

```
FaxManJr1.SendFax
```

```
End Sub
```

Be sure to alter the port number, file path, and FaxNumber to be valid values for your machine!

Your little sample app can now send a fax. The next step is to learn how to keep track of what your FaxMan Jr. control is doing.

Step 6: Add status reporting.

Double click on the FaxMan Jr. control. Add the following code to the Status event:

```
Private Sub FaxManJr1_Status(ByVal pStatObj As FaxmanJrCtl.IfaxStatusObj)  
Label1.Caption = pStatObj.CurrentStatusDesc  
End Sub
```

Your application will now report on FaxMan Jr.'s status throughout the faxing procedure.

Note that there are several other status messages that FaxMan Jr. sends to your application – these are all defined in the reference for the Status event.

Step 7: Seeing what FaxMan Jr. is doing.

Ok, this is the fun part. FaxMan Jr. actually sends your application the strings sent to and received from the faxmodem. By simply placing a little bit of code in the Message event, your application can display this data.

To see how, just double-click on the FaxMan Jr. control and select the Message event. Then add the following code to it:

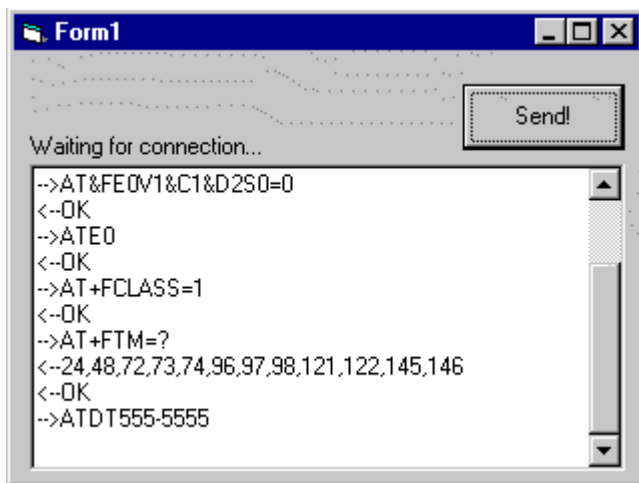
```
Private Sub FaxManJr1_Message(ByVal bsMsg As String, ByVal bNewLine As Integer)
```

```
Text1.SelText = bsMsg
```

```
If bNewLine Then Text1.SelText = vbCrLf
```

```
End Sub
```

If you've setup your textbox control properly, you should see something like this (this screenshot was taken while the fax was in-progress, so it still shows the "Waiting for connection" string):



Note that all of the strings FaxMan Jr. sends to the faxmodem are prefixed with "à", whereas all of the data returned from the faxmodem and read by FaxMan Jr. are prefixed with "ß".

That's Easy! What's Next?

Ok, now that you see how easy it is to use FaxMan Jr. to send a fax, you can either just start playing with it and rely on the reference section to help with the properties, methods, and events, or you can go ahead and tackle the in-depth tutorial presented in the next section.

1.5.2 FaxMan Jr. Tutorial

1.5.2.1 Getting Started – Initializing the FaxMan Jr. Control

The FaxMan Jr. control has several properties that you must consider before you can actually begin sending or receiving faxes. You would generally set these properties one time for each FaxMan Jr. control, then continually send or receive without altering them. Note that most of these properties default to a generally useful value – you just need to be aware of when you might want to alter these default values. The initialization properties are:

Port – sets the communications port that FaxMan Jr. will use to communicate with the faxmodem. This value is generally, but not limited to, 1 to 4 (Com1 to Com4), though it is possible to have more comm ports through the use of a multi-port serial card. Note that this value is not set to a default value – you must set the Port property before using FaxMan Jr. This implies that you or your application must ask the end-user which ports are installed, or that you somehow have knowledge of which comm ports have faxmodems on them – the FaxFinder control solves this problem if you haven't already determined the proper port through other means.

Class – Sets the class of the faxmodem to Class 1, Class 2, Class 2.0 or Class 2.1, as described earlier in the documentation in the faxmodem overview. This property defaults to Class 1, and this will work on 90% of the faxmodems available on the market. You would only need to change this property if you had a faxmodem that didn't support Class 1 or if you wanted to use one of the other device settings.

Reset – The Reset property is a string of characters that is used to reset the faxmodem to a known state in preparation for fax processing. FaxMan Jr. defaults to a generally acceptable value for this string. You may, depending on your application, have a special situation that requires a different Reset string.

Init – The Init string, like the Reset string, is a string of characters. It is used to initialize the faxmodem for faxing operations. As with the Reset string, FaxMan Jr. provides a generally acceptable default value.

1.5.3 FaxFinder

The FaxFinder control is a great help for those times where you need to determine what types of faxmodems are attached to a computer system. FaxFinder is a simple control that interrogates a system's comm ports and creates a collection of available devices. Your application can then go through the device collection returned from FaxFinder and select one of the devices directly or present a choice to your users.

FaxFinder has two properties:

DeviceCount – the number of faxmodems found connected to this machine. Note that the first time this property is accessed, FaxFinder takes several seconds to search for fax devices and stores the device descriptions. Subsequent accesses simply refer to this stored information.

Item – a collection of DeviceDesc objects. Each DeviceDesc object has the following properties:

Port – port number that this faxmodem is connected to

bClass1 – True if this faxmodem supports Class 1 instructions

bClass2 – True if this faxmodem supports Class 2 instructions

bClass20 – True if this faxmodem supports Class 2.0 instructions

bClass21 – True if this faxmodem supports Class 2.1 instructions

The code sample below illustrates how to use the FaxFinder. This example is a simple subroutine which fills a text control with a list of comm ports with installed faxmodems and the fax capabilities each supports:

```
Private Sub Command1_Click()
```

```
Dim i As Integer
```

```
Dim desc As DeviceDesc
```

```
For i = 1 To FaxFinder1.DeviceCount
```

```
Set desc = FaxFinder1(i - 1) 'This is a zero-based enumeration
```

```
Text1.SetText = "Port " & Str(desc.Port) & vbCrLf
```

```
Text1.SetText = "Supports: "
```

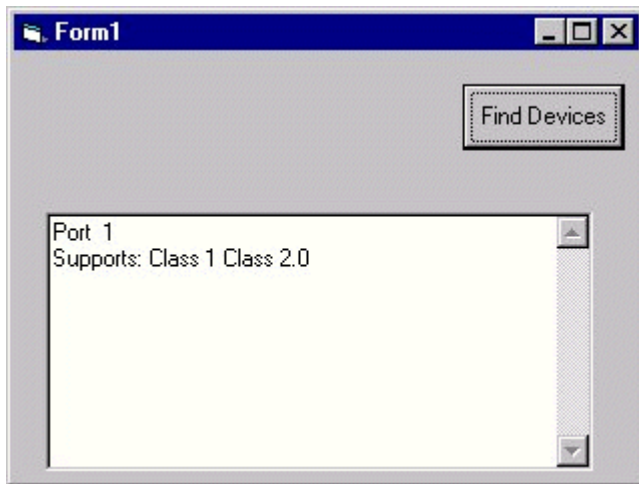
```
If desc.bClass1 Then Text1.SetText = "Class 1 "
```

```

If desc.bClass2 Then Text1.SelText = "Class 2 "
If desc.bClass20 Then Text1.SelText = "Class 2.0 "
Text1.SelText = vbCrLf & vbCrLf
Next i
End Sub

```

The following screen shows the results of running this subroutine:



1.5.4 Sending Faxes

Now that you've seen how to initialize a FaxMan Jr. control, it's time to start sending faxes. If you went over the quick start tutorial earlier in the documentation, you already realize how easy it really is to send a fax. The steps you have to take are these:

- You must have at least one page to fax. This means that you must either set the **FaxFiles (FaxFiles_Property.html)** Property or the **FaxCoverPage (FaxCoverPage_Property.html)** Property , or you can set both. (We cover how to create fax files in a later section). FaxMan Jr. allows you to send a virtually unlimited number of pages in a single fax job by chaining the list of files you wish to fax (For example, "c:\images\sales1.fmf+c:\images\sales5.fmf").
- Set the **FaxNumber (FaxNumber_Property.html)** to the phone number of the fax machine you wish to send to.

- Set the **Port (Port_Property.html)** property to the number of the comm port on which your faxmodem is located.
- Set any of the other properties that you might want to use. This is optional, but most applications will generally want to set things like the banner and cover page, plus all of the additional strings that can appear on banners or cover pages (name of sender, name of sender's company, subject, comments, etc...).
- Call the **SendFax (SendFax_Method.html)** method.

The concept is pretty simple, as you can see in the following example:

```
Private Sub Send_Click()
```

```
  `initialize the faxmodem settings
```

```
  faxctl1.Port = 1
```

```
  faxctl1.Class = 1
```

```
  `set the cover page and phone number
```

```
  faxctl1.FxCoverPage = "cover1.pg"
```

```
  faxctl1.FaxNumber = "555-1212"
```

```
  `now set any optional properties for the coverpage
```

```
  faxctl1.UserName = "Joe West" `name of person sending the fax
```

```
  faxctl1.UserCompany = "Jones & West, Inc." `name of company sending the fax
```

```
  faxctl1.UserFaxNumber = "555-3131" `fax number of sender
```

```
  faxctl1.UserVoiceNumber = "555-3130" `voice phone number of sender
```

```
  faxctl1.FaxName = "Jane Smith" `name of the person you're faxing to
```

```
  faxctl1.FaxSubject = "Notes from the meeting" `subject of the fax
```

```
  faxctl1.FaxCompany = "Smith Corp." `name of company you're sending the fax to
```

```
  `Comments can be as long as you want, just remember that they've got
```

```
  ` to fit on the cover page.
```

```
faxctl1.FaxComments = "Just thought you might want a heads up on the meeting  
we had today."
```

```
faxctl1.SendFax  
End Sub
```

You should note that all of these settings will be retained, so things like **UserName (UserName_Property.html)** and **UserCompany (UserCompany_Property.html)** can be set once and used over and over.

1.5.5 Creating Fax Files

The fax files used by FaxMan Jr. are multi-page files (i.e., a single file can contain an unlimited number of fax pages) which are stored in a TIFF Group 3 file .

It is important to note that FaxMan Jr. can accept any number of files in a single fax event. The files are simply "chained" together at schedule time by separating them with a "+" character, like this:

```
"cover.fmf+attach1.fmf+test.fmf"
```

This makes it extremely easy to keep your sales literature, for example, in separate attachment files (say one file for each product you sell) and then "chain" together a single fax event which contains information on all the products the client is interested in.

FaxMan Jr. can only fax files that are in a special file format. This file format, the FMF file format, is a TIFF Group3 file. There are basically only three ways to create FMF files:

1. Use the FaxMan Jr. Print Drivers and ActiveX control – by far the most common method, this allows users to generate faxable files from virtually any Windows application. Instructions on how to do this are located elsewhere in this manual.
2. Use the **ImportFiles (ImportFiles_Method.html)** Method to convert PDF*, monochrome BMP, PCX, DCX, or TIFF images (created by other methods) into FMF files. To do this, you simply pass the names of the files you wish to

convert to the ImportFiles Method, as follows :

```
Faxctl.ImportFiles "c:\faxes\temp.fmf", "c:\images\sample.tif+c:\images\test.pcx"
```

The first parameter is the output file you're trying to generate, the second parameter is the list of files you wish to convert.

All pages of multi-page TIFF, PDF or DCX image files will be converted for you. FaxMan Jr. only converts monochrome images. If the image is a color image or has an associated palette (an image could be a 256 color BMP with only black and white pixels on the image) the image will need to be dithered down to black and white. ImageMan can provide this functionality as well as much more, see below. If the image is less than a full page it will be padded with white space to the right of the image.

3. Use DTI's ImageMan product to convert any of the raster file formats ImageMan supports, including the ability to scale, color reduce, and dither images. When converting any image to an FMF file using ImageMan the following steps need to be taken into consideration: The file needs to be reduced to a monochrome image. The image will need to be scaled or cropped to 1728 pixels wide. Using the SaveOptions settings the TIFF_XRES will need to equal 200 and the TIFF_YRES will need to be set to 98 for low resolution faxes or 196 for high-resolution faxes. The image length is up to the user but for an 11-inch long page at high resolution the height will need to be set to 2156. Fillorder=2 (reverse byte order), and either single or multi strip is OK.

* Requires optional PDF Import Add-on. For more information visit our **website** (<http://www.data-tech.com.com/content/faxpdf.aspx>).

1.5.6 Fax Resolution

y

Faxing technology allows any fax to be sent in either high or low resolution. FaxMan Jr. allows your application to select between high and low resolution quite easily by setting the **FaxResolution (FaxResolution_Property.html)** Property to 0 for low resolution and 1 for high resolution. Simple, right?

Well, not quite so simple. The thing to keep in mind is that the files you're trying to fax may be stored in a different resolution from the resolution you're trying to send them at. This means that FaxMan Jr. may have to stretch or compress your files to send them at the resolution you're asking for. The practical result of this is

that images you've created at low resolution and sent at high resolution may look somewhat blockier than you think they should. Likewise, high-resolution images sent at low resolution may be missing lines, especially on detailed images or small text.

Another problem is that the fax device you're sending to may not allow sending in high resolution. In this case, the fax will be sent at low resolution regardless of your request to send at high resolution. This case is rare, as most modern devices can readily handle high-resolution images, but it can happen.

1.5.7 Sending Options

There are several properties that can be set before sending a fax. These properties have various effects, all of which are pretty self-explanatory. For more information, check out the reference section for the FaxMan Jr. control. The complete list of properties that affect your fax in some way (as opposed to "informational" items as used on banners and cover pages) is given below:

FaxBanner (FaxBanner_Property.html) – adds a banner to each fax page

FaxCoverPage (FaxCoverPage_Property.html) – adds a cover page to the fax

FaxFiles (FaxFiles_Property.html)– specifies the list of files to fax

FaxNumber (FaxNumber_Property.html) – the phone number of the fax recipient

FaxResolution (FaxResolution_Property.html) – the desired resolution to use when sending the fax

1.5.8 Receiving Faxes

Receiving faxes with FaxMan Jr. is just as simple as sending faxes, though you have less to do. Basically, all you do is to initialize a few properties and call the Receive method with the number of rings you want to see before answering.

A quick example is shown below:

```
Private Sub Receive_Click()  
    faxctl1.Port = 1  
    faxctl1.Class = FAX_1 `use Class 1
```

```
faxctl1.Receive(2) `answer after 2 rings
```

```
End Sub
```

That's pretty much all there is to it. FaxMan Jr. will attempt to receive a fax, firing notification events along the way, and will store the received fax file for you.

There are a couple of properties that can affect the receive process. These are as follows:

ReceiveDir Property (ReceiveDir_Property.html) – This is the directory where you want FaxMan Jr. to store received fax files. FaxMan Jr. defaults to storing faxes in the same directory that the FaxMan Jr. control is located, but through this property you can override that behavior. Note that if the directory you set is not valid, FaxMan Jr. will place received faxes in the default location.

Local ID Property (Local_ID_Property.html) – Setting this property before receiving allows your faxmodem to identify itself to callers. You can use up to 20 characters (more than this will be truncated).

Note that it is possible to set the number of rings to zero in order to begin receiving immediately. This can be useful in certain circumstances.

1.5.9 Event and Status Objects

It's great that FaxMan Jr. makes sending faxes so easy...but how the heck does your application have any idea what FaxMan Jr. is doing at any given time? The answer to this, of course, is FaxMan Jr. events and the Status Object.

FaxMan Jr. fires events in your application at various milestones during the sending process (and receiving process, as you'll see later). These events are as follows:

Status (Status_Event.html) – fired during the sending (or receiving) process to give your app up to the second information on the current send or receive operation.

NegotiatedParms (NegotiatedParms_Event.html) – fired when the fax devices have completed negotiations for the current fax session. The RemoteID is

also valid at this stage of the process.

Pages (Pages_Event.html) – notifies your app of the number of pages sent/received.

StartTime (StartTime_Event.html) – fired when the fax process is begun.

EndTime (Endtime_Event.html) – fired when the fax process ends.

CompletionStatus (Completion_Status_Event.html) – fired at the end of the faxing process to indicate that your app can check the status object to determine the overall success of the fax process.

Message (Message_Event.html) – fired for messages sent to and received from the faxmodem. Also used to output debug messages during sending/receiving. Your app can choose to monitor these strings or may just disregard them.

Ring (Ring_Event.html) – fired when listening for rings. Your app can choose to answer the ring or simply continue listening.

ReceiveFileName (Receive_Filename_Event.html) – fired during fax reception when the receive filename has been created.

In addition, FaxMan Jr. keeps track of the current state of the faxing process in an object called the Status object. This object is passed to several FaxMan Jr. notification events so that your application can determine FaxMan Jr. status at these times. The Status Object has the following members:

RemoteID (Status_Object_RemoteID_Property.html) – a string that identifies the remote fax device.

ConnectSpeed (Status_Object_ConnectSpeed_Property.html) – an Integer containing the negotiated speed of the faxing process (300 – 14400).

NegotiatedResolution

(Status_Object_NegotiatedResolution_Property.html) – an Integer containing the negotiated resolution of the fax process (0 for low, 1 for high).

CurrentStatus (Status_Object_CurrentStatus_Property.html) – an Integer which contains the current status of a fax send/receive operation.

CurrentStatusDesc (CurrentStatusDesc_Property.html) – a String which contains the current status of a fax send/receive operation.

StartTime (Status_Object_StartTime_Property.html) – a DWORD containing the start time of the current fax operation (i.e., when the phone is taken off-hook). This is expressed in seconds elapsed since midnight, January 1, 1970.

EndTime (Status_Object_EndTime_Property.html) – like StartTime, but specifies the time a fax process ended (i.e., when the phone is hung up).

ReceiveFileName (Status_Object_ReceiveFileName_Property.html) – a string that identifies the name of the received fax file.

Pages (Status_Object_Pages_Property.html) – Integer which specifies the total number of pages in the current fax operation.

PagesCompleted (Status_Object_PagesCompleted_Property.html) – an Integer which indicates the number of pages successfully completed in the current fax operation.

ErrorCode (Status_Object_ErrorCode_Property.html) – an Integer which specifies the current error status.

ErrorDesc (Status_Object_ErrorDesc_Property.html) – a string which describes the current error status.

LastEventType (Status_Object_LastEventType_Property.html) – an Integer which identifies which type of event was last fired. This is normally used only for applications which cannot process events, and which must use the WaitForEvent method to watch events.

The Status Object is passed to the following events:

Status (Status_Event.html)

NegotiatedParms (NegoitatedParms_Event.html)

Pages (Pages_Event.html)

StartTime (StartTime_Event.html)

EndTime (Endtime_Event.html)

CompletionStatus (Completion_Status_Event.html)

ReceiveFileName (Receive_FileName_Event.html)

You may refer to any of the Status Object's properties from within any of these event handlers. The example below is an excerpt from an application and illustrates how the Status Object is used in several event handlers:

```
Private Sub faxctl1_CompletionStatus(ByVal pStatObj As
FaxmanJrCtl.IfaxStatusObj)
ErrStat.Caption = "Error Status: " + Str$(pStatObj.ErrorCode)
End Sub
```

```
Private Sub faxctl1_Pages(ByVal pStatObj As FaxmanJrCtl.IfaxStatusObj)
Pages.Caption = "Pages: " & Str(pStatObj.PagesCompleted) & " of " & Str
(pStatObj.Pages)
Text1.SelText = "Pages: " & Str(pStatObj.PagesCompleted) & " of " & Str
(pStatObj.Pages) & vbCrLf
End Sub
```

```
Private Sub faxctl1_ReceiveFileName(ByVal pStatObj As
FaxmanJrCtl.IfaxStatusObj)
Text1.SelText = "Receive File Name: " & pStatObj.ReceiveFileName & vbCrLf
ErrStat.Caption = "Receive File Name: " & pStatObj.ReceiveFileName
End Sub
```

```
Private Sub faxctl1_NegotiatedParms(ByVal pStatObj As
FaxmanJrCtl.IfaxStatusObj)
Speed.Caption = pStatObj.ConnectSpeed
If pStatObj.NegotiatedResolution = 0 Then
Resolution.Caption = "Low"
Else
```

```

Resolution.Caption = "High"
End If
RemoteID.Caption = pStatObj.RemoteID
End Sub

```

```

Private Sub faxctl1_Status(ByVal pStatObj As FaxmanJrCtl.IfaxStatusObj)

```

```

Text1.SelText = "Status: " + Str(pStatObj.CurrentStatus) + vbCrLf
Status.Caption = pStatObj.CurrentStatusDesc

```

```

Status.Refresh
End Sub

```

You can find information on individual event callback functions in the reference section.

1.5.10 Rings

FaxMan Jr. also gives you the ability to simply listen for a ring on the phone line connected to the faxmodem by calling the Listen method. The Listen method waits until a ring is detected on the phone line. When a ring is detected, the Ring event is fired, and your application gets an opportunity to decide what happens next, depending on the value you set for the Action parameter passed to the Ring event. These are the valid values for the Action parameter:

Action = 0 – quit listening for rings and return from the Listen method.

Action = 1 – ignore the ring and continue listening for rings.

Action = 2 – answer the ring and attempt to receive a fax.

The following Ring event code listens for 3 rings and then attempts to receive a fax:

\nRingCnt is a global variable which is set to 0 the first time

\this event is fired...

```

Private Sub faxctl1_Ring(pnAction As Integer)

```

```

nRingCnt = nRingCnt + 1

if nRingCnt = 3 then
  nRingCnt = 0 `reset this for next calls
  pnAction = 2 `now try to receive a fax
else
  pnAction = 1 `keep listening `till we get 3 rings
end if
End Sub

```

1.5.11 Cancelling Faxes

There are times when you will need to cancel a fax operation that is currently in progress. This may be required for a variety of reasons, such as wanting to terminate a Listen method call without having to wait for a ring, wanting to shut down your application while it's waiting to receive a fax, and so on. Canceling an operation in this manner is quite easy – simply call the **CancelFax** Method.

Canceling a fax operation can take anywhere from a few milliseconds to several seconds (sometimes up to 10 seconds) to complete, and during this time your application is still active, so you should be sure to wait until you see the **CompletionStatus** event fired before you exit your application or attempt to start another fax operation.

1.6 Using the .Net Components

1.6.1 Quick Start

Sending Faxes using the Jr .Net components involves three simple steps:

1. Creating the files to be faxed

Faxable Files can be created by importing TIFF, BMP, TXT or PDF* files, by creating TIFF files with the appropriate options using our ImageMan.Net toolkit or

some other imaging toolkit or by using an instance of the FaxMan virtual printer driver to create a fax file from any application that supports printing.

2. Setting the appropriate properties in the Fax object.

After creating an instance of a Fax object some properties must be set including the **FaxNumber** and **CoverPage** or **Files** properties. Most applications will set more but this is the minimum set of properties that must be set to be able to send a fax. The files specified by the **Files** property must already be in faxable format.

3. Sending the fax using the Send method of the Faxing object.

After creating an instance of the **Faxing** object, the application must set the **Modem** property to a **Modem** object obtained either from the **Modems** collection or created with the appropriate values. The Modem object specifies the comm port to use and also which Fax class will be used to send the fax.

The **Send** method can then be called to initiate sending the fax. During the sending process the Faxing object will generate several events including the **FaxStatus**, **FaxMessage**, **FaxSetStartTime**, **FaxSetEndTime** and **FaxNegotiatedParameters** to signal the application of the current status of the faxing process. The event handlers receive a **FaxEventArgs** object which contains a **Fax** object containing the latest state of the job.

Once the fax has completed sending the control will fire the **FaxCompletionStatus** event which indicates the job has completed.

 * PDF Import requires the optional FaxMan PDF Import Addon component

1.6.2 Using the Modems Collection

The Modems class makes it simple to find out what faxmodems are installed on a PC. Calling the **AutoDetect** method will cause the component to iterate thru the installed modems on the system and populate the Modems object with a collection of Modem objects describing the modems it found.

The **Modem** object contains properties which describe each modem including what classes are supported, the commport and the init and reset strings.

The **Class1**, **Class2**, **Class20** and **Class21** properties return true if the modem supports that Fax Class. The **FaxClass** property is set to the Fax Class that will be used to communicate with the modem. FaxMan Jr will set this to default value based on which classes the modem supports. Your application can override this default by setting this property to different class prior to setting the **Modem** property of the **Faxing** object. Note that if your application sets this to a class not supported by the modem then faxing will fail.

The **Port** property returns the Comm Port ID of the selected modem.

The **Init** string contains the initialization string that will be sent to the modem after sending the **Reset** string. The FaxMan Jr default string will work for most modems, though this can be changed should you need to add additional commands for your modem.

The **Reset** string is sent to the modem after the comm port is opened and would rarely need to be changed from the default.

// Sample Code for enumerating thru all the modems on a system

1.6.3 Creating Fax Files

FaxMan Jr Fax files are standard Group3 compressed multi page tiff files with a required set of TIFF tags and attributes. There are three basic ways to create files you can fax with FaxMan Jr:

1. **Using the FaxMan Jr printer driver**

This is the most common method of creating files and also the simplest for most applications. The FaxMan virtual printer driver can be installed programmatically by your application using the **Printer Control** or can be installed as part of your application's install. Once installed the driver takes print output and creates a faxable file and notifies your application via the **PrintComplete** event of the Printer control.

2. **Importing TIFF, BMP, TXT and PDF* Files**


Using the **ImportFiles** method of the Faxing class your application can import black & white (1 bit) TIFF, BMP, DCX, PCX files as well as TXT and optionally PDF files.

3. **Creating TIFF files directly**

You can use an imaging toolkit such as our ImageMan.Net toolkit or others to create TIFF files with the proper attributes. Basically the TIFF file must meet

the following specifications:

- 1 Bit
- 1728 Pixels in width
- 196 or 98 DPI vertical resolution
- Group3 Compressed with Byte Aligned EOLs
- FillOrder set to 2

 * The FaxMan PDF Import Addon is an optional, extra cost addon. The FaxMan Jr install includes a trial version of this option allowing you to test the PDF Importer.

1.6.4 Sending Faxes

Sending a fax involves creating an instance of the **Fax** class and setting the desired properties in that object then calling the **Send** method of the **Faxing** object to start sending the fax.

The Fax object contains all the information about the fax including the files to sent, coverpage information such as Sender's Name, Recipients Name, etc.

Faxes can consist of only a coverpage or of one or more fax files optionally including a cover page. Either the CoverPage or FaxFiles properties must be set to valid values prior to sending the fax. FaxMan Jr supports cover pages in our own .PG format and the .CPE Windows 9x Cover page format. Fax files must be converted to the FaxMan Jr FMF file format prior to sending. This file format is basically a multi page Group3 compressed TIFF file which has been written with certain TIFF tags. Tiff files created by other applications will generally need to be imported using the **ImportFiles** method to be sure they are written with the proper parameters. It is possible to write a tiff file from another application so its directly faxable but it requires the file be written with a strict set of tiff tags.

The minimum Fax object properties required to send a fax are:

- **CoverPage** or **Files** - One or both of these properties must be set.
- **FaxNumber** - The phone number to dial including any prefix need to get an outside line

The minimum Faxing object properties that must be set are:

- **Modem** - Set to a Modem object specifying the modem and fax class to use for sending

Once the appropriate properties have been set the fax can be sent by calling the **Send** method of the **Faxing** class and passing the Fax object.

Control will return to your application after calling **Send** and the fax will be sent in a background thread. While the fax is being sent the Faxing class will fire several events indicating the progress of the fax being sent. Your application can expect to receive the following events:

- **FaxSetStartTime** - Fired when the remote fax machine has connected.
- **FaxStatus** - Fired periodically as each page is sent so you can update a percentage of page complete type counter
- **FaxSetPages** - Fired after each page has been sent to allow your application to update a pages sent counter
- **FaxNegotiatedParameters** - Fired when the fax parameters have been negotiated between the two fax devices.
- **FaxSetEndTime** - Fired when the fax job has been completed.
- **FaxCompletionStatus** - Fired at the very end of the fax send process, this is the last event your application will receive when sending and is used to determine if the fax was sent successfully.

Each of these events includes a **FaxEventArgs** object passed to the event handler, this object contains a **Fax** object containing the latest status of the fax.

Once the **FaxCompletionStatus** event has fired your application can use the Faxing object to send or receive another fax. Calling the Send, Receive or Listen methods while the control is still busy sending a fax will cause an exception to be generated.

Send A Coverpage in C#

```
Faxing faxman = new Faxing();
Fax fx = new Fax();
fx.FaxNumber = "1234";
fx.Coverpage = @"c:\cover1.pg";
fx.Comments = "This is my first fax";
faxman.Modem = new Modem(3, FaxClass.Class20); // Use COMM3 and Class 2.0
faxman.Send( fx );
```

Send A Coverpage in VB.Net

```
Dim faxman As Faxing = New Faxing()
Dim fx As Fax = New Fax()

fx.FaxNumber = "1234"
fx.Coverpage = "c:\cover1.pg"
fx.Comments = "This is my first fax"

faxman.Modem = New Modem(3, FaxClass.Class20) ' Use COMM3 and Class 2.0
faxman.Send(fx)
```

1.6.5 Receiving Faxes

Receiving faxes with the FaxMan Jr Faxing class is very easy.

1. **Create an instance of the Faxing control**
2. **Set the Modem property to the modem you wish to use**
You can set the **Modem** property to a Modem object from the **Modems** collection or by creating a Modem object manually.
3. **Set the ReceiveDir property to the directory where received faxes should be stored**
This directory should exist and your application must have read/write/modify permissions in that directory.
4. **Use the Receive method or the Listen method and FaxRing Event.**
Both options allow you to receive a fax but with slightly different options. When your application calls the **Receive** method it passes the number of rings to wait before answering and control is returned to your application. The FaxMan Jr control is listening for a call on a background thread and will start to fire events when a call comes in. The **FaxSetStartTime**, **FaxSetPages**, **FaxStatus**, **FaxReceiveFileName**, **FaxNegotiatedParameters** and **FaxSetEndTime** events will fire while the fax is being received and the **FaxCompletionStatus** event will fire at the end of the fax indicating if it was received successfully or not.

Calling the **Listen** method will cause the control to listen for Rings from the modem and fire a **FaxRing** event when it gets one. Your application can then set the **Action** property of the **FaxRingArgs** object to pickup the call and begin receiving or to stop listening. If your app sets the Action to **ReceiveFax** then FaxMan will pickup the call and start receiving and firing the same events as if the **Receive** method had been called.

The received fax will be stored in a multi page tiff file in the directory specified by

the **ReceiveDir** property. The name is automatically generated by FaxMan and will be returned to your application in the FaxReceiveFileName event and also in the Fax object passed to the **FaxCompletionStatus** event. Your application can rename the file any time after the **FaxCompletionStatus** event has fired.

Because received files may have some image coding inconsistencies which may cause some image viewers to fail to display the image, your application can use the **ImportFiles** method to clean the received file. This can be done in your **FaxCompletionStatus** event handler.

Receiving a Fax with C#

```
Faxing faxman = new Faxing();
```

```
faxman.Modem = new Modem(3, FaxClass.Class20); // Use COMM3 and Class 2.0
faxman.Receive( 2 ); //Wait 2 Rings before picking up
```

Receiving a Fax with VB.Net

```
Dim faxman As Faxing = New Faxing()
```

```
faxman.Modem = New Modem(3, FaxClass.Class20)      ' Use COMM3 and Class
2.0
faxman.Receive(2)      ' Wait 2 Rings before picking up
```

1.6.6 Viewing Faxes

The .NET Framework has support for viewing tiff files. It is relatively intolerant code though. If it runs into an error on one line it will fail to display the rest of the page painting it solid instead. The fax files we receive are written straight to file as received from the modem. This presents a problem with the .NET framework. One solution is to simply import the file using the **ImportFiles** method of the Faxing object. This works in most cases.

Another option for viewing files is the ImageMan.NET components, this obviates the need to import the received files. It also provides a host of useful features which improve user experience. The ImageMan.NET has exceptional viewing code for viewing black and white code this feature is called antialiasing. It also allows you to view the image at it's native bit depth(black and white) instead of having to inflate it to 32 times it's original size in memory as the framework does in order to display it with GDI+. In addition to better scaling and viewing there is a thumbnail control for viewing all the pages in a received fax, image preview dialogs for loading faxes, and rotation as well as deskew for faxes that come in upside down or did not feed correctly. ImageMan.NET also has full support for working with memory streams as well, so you can write your faxes to a database

and read them without having to write them to disk.

.NET Framework code to load all the pages of a fax into an array for displaying each page one at a time.

```
using System.Drawing;
```

```
// Form level variable
private ArrayList imgs
```

```
private ArrayList LoadFaxFile(String fileName)
{
    ArrayList imgs = new ArrayList();

    // load image into bitmap ArrayList
    // Sets the tiff file as an image object.
    System.Drawing.Image objImage = Image.FromFile(fileName);
    Guid objGuid = objImage.FrameDimensionsList[0];
    System.Drawing.Imaging.FrameDimension objDimension = new
    System.Drawing.Imaging.FrameDimension(objGuid);

    // Gets the total number of frames in the .tiff file
    int totFrame = objImage.GetFrameCount(objDimension);

    // Saves every frame as a separate image in the ArrayList
    int i;
    for (i = 0; i < totFrame; i++)
    {
        objImage.SelectActiveFrame(objDimension, i);
        Image img = (System.Drawing.Image)objImage.Clone();
        imgs.Add(img);
    }

    return(imgs);
}
```

In the Main FaxMan Jr .NET samples we demonstrate using the Framework to display the image. We have put a panel on the form and set its AutoScroll property to true;

```
private System.Windows.Forms.Panel pnViewFax;
pnViewFax.AutoScroll = true;
```

We use this panel to host the PictureBox control, by dropping the PictureBox on to the panel in the forms designer.

```
this.pnViewFax.Controls.Add(this.pictureBox1);
```

This is the code to load the image from the array into the picture box. Note that once the picture is loaded the picture box is much bigger than the form. Since the panel has AutoScroll set to true scroll bars appear after we load the image allowing us to scroll around the image.

```
private void LoadImage() { LoadImage(0); }
private void LoadImage(int page)
{
    if ( (null != imgs) && (imgs.Count > page) )
    {
        pictureBox1.Image = (System.Drawing.Image)imgs[page];
        pictureBox1.Width = pictureBox1.Image.Width;
        pictureBox1.Height = pictureBox1.Image.Height;
        pictureBox1.Refresh();
    }
}

private void OpenFaxFile(String faxFile)
{
    if (faxFile.Length > 1)
    {
        imgs = LoadFaxFile(faxFile);
        LoadImage();
    }
}
}
```

Finally we want to be able to page through the image so we have added some menu handlers.

```
// Form Level Variable
private int _page;

private void mnuPageFirst_Click(object sender, System.EventArgs e)
{
    LoadImage(_page = 0);
}

private void mnuPagePrevious_Click(object sender, System.EventArgs e)
{
    if (_page > 0) LoadImage(--_page);
}

private void mnuPageNext_Click(object sender, System.EventArgs e)
{
    if (imgs.Count - 1 > _page) LoadImage(++_page);
}
}
```

```
private void mnuPageLast_Click(object sender, System.EventArgs e)
{
    if (imgs.Count - 1 > _page) LoadImage(_page = imgs.Count - 1);
}
```

The ImageMan .NET controls already have a concept of an image collection. Making it even easier to work with the images individually or as a group - scaling to fit the window or the width of the window, rotating, rubber band zoom, panning, page scrolling that responds to the mouse wheel, saving, printing and more.

1.6.7 Using the Printer Control

The FaxMan Jr printer control allows your application to create a virtual FaxMan printer driver for creating fax files from print jobs. The control allows your application to see if a printer is installed, to install a printer instance and to get notifications from its associated printer that a print job has completed.

1. **Instantiate the Printer control**

2. **Set the PrinterName property**

The **PrinterName** property should be set to the the name of the printer as you want it to appear in the Printer's folder.

3. **Installing the printer if needed**

Once the **PrinterName** property is set, your application can query the **IsPrinterInstalled** property to see if that printer has already been installed, if not then the application can use the control to install it.

Before calling the **InstallPrinter** method the application can set the **PrintFilePath** to specify the directory in which fax files will be created. **InstallPrinter** can then be called passing the path and name of the exe to be associated with the driver. The exe specified here is used to startup the application when a print job is sent to the printer and the application isn't already running.

4. **Handle the PrintComplete event**

When a print job is finished, the control will fire the **PrintComplete** event. Your application should call the **GetNextPrintJob** method of the **PrintJobs** collection which is returned in the **PrintJobs** property of the printer control. The **PrintJobs** collection is a FIFO list of all print jobs that have been completed and not previously retrieved. The **GetNextPrintJob** method returns a **PrintJob** object which contains properties for **FileName**, **DocumentName**, **Pages** and **Status** of the print job.



Installing the printer driver requires Administrative rights on most operating systems. The InstallPrinter method will fail in the event the user doesn't have the appropriate rights.

Due to the security on Vista applications run under a limited rights scenario even if the user is logged in as an Administrative user. To run with Administrative rights the application must either be started by right clicking on the icon and selecting 'Run As Admin' or by embedding a Vista Manifest in the executable. In both cases Vista will still display a UAC dialog confirming the user wants to run the application as Admin.

To run the command line as administrator:

Click on Start or press the Windows Key, type cmd, but this time, don't press Enter, instead, press CTRL+SHIFT+Enter. You'll be prompted to approve the action, if successful the dialog title will start with "Administrator:"

For this reason we recommend that if you plan to support Vista that you install your FaxMan printer driver instance from your application's installer which should already be running as Admin. This way your application can run under the limited user account and the printer will already be installed.

Using the Printer Control in C#

```
FaxPrint print;
...
print = new FaxPrint();
print.PrinterName = "My Fax Printer";
if( !print.IsPrinterInstalled )
print.InstallPrinter( Application.ExecutablePath );
print.PrintComplete += new
DataTech.FaxManNet.FaxPrint.__Delegate_PrintComplete(print_PrintComplete);
...
private void print_PrintComplete()
{
    PrintJob job = print.PrintJobs.GetNextPrintJob();
    string file = job.FileName;
    // The faxfile has been created and can be sent now
```

```
}

```

Using the Printer Control in VB.Net

```
Dim print As FaxPrint
...
print = New FaxPrint()
print.PrinterName = "My Fax Printer"
If (Not print.IsPrinterInstalled) Then
    print.InstallPrinter(Application.ExecutablePath)
End If
print.PrintComplete += New
DataTech.FaxManNet.FaxPrint.__Delegate_PrintComplete(print_PrintComplete)
...
Private Sub print_PrintComplete()
    Dim job As PrintJob = print.PrintJobs.GetNextPrintJob()
    Dim file As String = job.FileName
    ' The faxfile has been created and can be sent now
End Sub

```

1.6.8 Batch Sending of Faxes

Many sending applications are automated and have minimal user interaction. The user interface may be limited to a configuration utility or may be absent entirely. The faxes may come out of a database table or other source. There may be more faxes that need to be sent at a given time than there are com ports. This is the kind of scenario that the FaxMan SDK excels at. Most developers who are writing Client Server Applications or broadcast fax applications will choose the FaxMan SDK because it includes scheduling logic, automatic retries, and dynamically handles a pool of modems with no need to add or modify code on the application developers part.

Having explained the virtues of the FaxMan SDK I will go on to say that FaxMan Jr can also be used for writing a fax broadcast application it just takes more work on the part of the application developer. I will not attempt to recreate all of FaxMan SDK's features in the code sample below. This is not to say that writing modem pooling and scheduling logic aren't important- just that they are outside the scope of this example. Instead I will focus only on how to send multiple faxes on one modem.

The key to keeping one port busy is to watch for the `CompletionStatus`. This event gets fired each time a fax operation is done, whether successful or not. This is the event to schedule the next fax in. For this sample I will use a simple queue for holding multiple Fax objects. In your application you might be polling your database on a timer, in the `CompletionStatus`, or it may be run on demand by the System or DB Scheduler. The fax information from your database can then be supplied to the Fax object from one or more tables.

```
// Form Level Variables
DataTech.FaxManJr.Faxing _faxing;
System.Collections.Queue _faxJobs;

// Somewhere in your startup code choose your modem
...
_faxing.Modem = new Modem(3, FaxClass.Class1);
...

private void mnuQueueUp100Faxes_Click(object sender, System.EventArgs e)
{
    for (int i = 0; i < 100; i++)
    {
        DataTech.FaxManJr.Fax fax = new DataTech.FaxManJr.Fax();

        // Use report generating tool to generate pdf file bypassing printer driver
        string myPdfReport = "c:\\myPdfReport.pdf";

        // Generate Fax File from pdf file using Pdf AddOn
        // Give the fmf file a unique name unless your report names are already
        unique
        string importedfile = String.Format("{0}{1}{2}.fmf",
            Application.UserAppDataPath,
            myPdfReport.Substring(myPdfReport.LastIndexOf("\\"),
            myPdfReport.Length - myPdfReport.LastIndexOf("\\")), i);
        if (! (System.IO.File.Exists(importedfile)) )
            _faxing.ImportFiles(myPdfReport, importedfile);

        // Get fax information from database
        fax.FaxNumber = @"123-4567";
        fax.FaxFiles.Add(importedfile);
        fax.UserData = String.Format("Fax {0}", i);
        _faxJobs.Enqueue(fax);
    }
    DataTech.FaxManJr.Fax qFax = (DataTech.FaxManJr.Fax)_faxJobs.Dequeue();
    _faxing.Send(qFax);
}

// This is the event where the modem is ready to send the next fax in the Queue
private void _faxing_FaxCompletionStatus(object sender, FaxEventArgs args)
{
```

```

// reschedule if failed the first time
// by sending to end of queue
if ( (FaxError.Ok != args.Fax.FaxError) && (args.Fax.UserData.Length < 10) )
{
    args.Fax.UserData = String.Format("Failed To Send Fax {0}, Trying Again",
args.Fax.UserData);
    _faxJobs.Enqueue(args.Fax);
}

// Send the next fax in the queue
if (_faxJobs.Count > 0)
{
    DataTech.FaxManJr.Fax qFax = (DataTech.FaxManJr.Fax)_faxJobs.Dequeue
());
    _faxing.Send(qFax);
}
}

// And finally while testing you probably want // to be able to cancel this
operation.
private void mnuStopBatchSend_Click(object sender, System.EventArgs e)
{
    _faxJobs.Clear();
    _faxing.CancelFax();
}

```

The main sample also demonstrates batch sending of faxes under the test menu. The main sample as well as this sample send the same fax to the same number so are limited in scope. But with supplying the key information from your database like phone number and the file you want sent as well as any banner or cover page information you should at least have a good start. And finally if you want advanced scheduling options including modem pools consider the FaxMan SDK.

1.6.9 AutoReceiving of Faxes

Implementing Automatic Receiving of Faxes in Your Application

The Main .NET samples implement an Auto Receive function, how can I implement this in my application?

The FaxCompletionStatus is the key to this functionality. When this event is fired the fax modem is now ready to either listen for receiving another fax or send a fax that may be queued up.

If all this modem does in your application is to receive faxes then this event is very simple to implement. Just tell it to prepare to receive another fax.

C# Receive Sample

```

// Form Level Variables
DataTech.FaxManJr.Faxing _faxing;

// This is the event when the device is no longer busy.
private void _faxing_FaxCompletionStatus(object sender, FaxEventArgs args)
{
    // Did my last received fax come in successfully?
    if (DataTech.FaxManJr.FaxError.Ok == args.Fax.FaxError)
        // Do something with the fax
        System.Diagnostics.Debug.WriteLine(
            String.Format("Successfully received fax file: {0}",
                args.Fax.FaxFiles[0].ToString()));

    // Ready to Receive another fax
    _faxing.Receive(1);
}

// And make sure you start the receiving process somewhere else in your
// application:
private void Form1_Load(object sender, System.EventArgs e)
{
    // Set the modem up for use
    _faxing.Modem = new Modem(3, FaxClass.Class21);

    // The single parameter tells FaxMan Jr to
    // let the phone ring once before picking up.
    _faxing.Receive(1);
}

```

VB.Net Receive Sample

```

' Form Level Variables
Private _faxing As DataTech.FaxManJr.Faxing
' This is the event when the device is no longer busy.
Private Sub _faxing_FaxCompletionStatusEvent(ByVal sender As Object, ByVal a
FaxEventArgs)
    ' Did my last received fax come in successfully?
    If DataTech.FaxManJr.FaxError.Ok = args.Fax.FaxError Then
        ' Do something with the fax
        System.Diagnostics.Debug.WriteLine(String.Format("Successfully received fi
file: {0}", args.Fax.FaxFiles(0).ToString()))
    End If
    ' Ready to Receive another fax
    _faxing.Receive(1)
End Sub
' And make sure you start the receiving process somewhere else in your
// application:

```

```

Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    ' Set the modem up for use
    _faxing.Modem = New Modem(3, True, False, False)
    ' The single parameter tells FaxMan Jr to
    ' let the phone ring once before picking up.
    _faxing.Receive(1)
End Sub

```

The samples do more than this. They use the line for both sending and receiving. When the modem is not busy sending it will revert to receive mode. When a user wants to send a fax and if the modem is not currently receiving a fax it will stop listening for an incoming fax and send out the fax.

So how does this look in my application?

C# Sample

```

// Form Level Variables
DataTech.FaxManJr.Faxing _faxing;
DataTech.FaxManJr.Fax _fax;
bool _waitingToSend;

// We will have the modem ready to receive
// for the entire time the application is running
private void Form1_Load(object sender, System.EventArgs e)
{
    // Modem on port 2 is a USB modem so I am passing "&F1 instead
    // of the default "&F" into the init string so that flow control is enable
    // I am also passing "M0" to turn of the modems speaker.
    _faxing.Modem = new Modem(2, FaxClass.Class20, "AT&F1E0V1&C1&D2S0=0M0", "A
    _faxing.Receive(1);
}

private void mnuSend_Click(object sender, System.EventArgs e)
{
    // Fill in _fax object with sending information
    // The comma in the phone number tells the modem
    // to pause to get an outside line on a pbx
    _fax.FaxNumber = @"9,1-123-456-7890";
    _fax.FaxFiles.Add(@"c:\myfaxfile.fmf");
    _fax.UserData = "sending fax";

    if (_faxing.DeviceStatus == DeviceStatuses.DeviceReceiveMode)
    {
        _faxing.CancelFax();
        _waitingToSend = true;
    }
    else if (_faxing.DeviceStatus == DeviceStatuses.DeviceCurrentlyReceiving)
    {

```

```

        // Don't cancel fax in the middle of reception, simply wait.
        _waitingToSend = true;
    }
    // In this sample we are not going to let someone schedule
    // while they are currently sending.
    // See Batch Sending example for queueing multiple faxes
    else
    {
        System.Windows.Forms.MessageBox.Show("You are currently sending a fax,
please wait.",
        "My Sample App");
    }
}

// This is the critical event where we decide what to do next
private void _faxing_FaxCompletionStatus(object sender, FaxEventArgs args)
{
    // First what happened with the last fax
    if (args.Fax.UserData == "sending fax")
    {
        if (DataTech.FaxManJr.FaxError.Ok == args.Fax.FaxError)
            System.Diagnostics.Debug.WriteLine("Successfully sent your last fax");
        else
            System.Diagnostics.Debug.WriteLine("There were problems sending your
last fax!");
    }
    else if (DataTech.FaxManJr.FaxError.Ok == args.Fax.FaxError)
        System.Diagnostics.Debug.WriteLine(
            String.Format("Successfully received fax file: {0}",
            args.Fax.FaxFiles[0].ToString()));

    if (_waitingToSend)
    {
        // Let's send this fax that is ready to go out
        _waitingToSend = false;
        _faxing.Send(_fax);
    }
    else
    {
        // Ready to Receive another fax
        _faxing.Receive(1);
    }
}
}

```

An important addition to the FaxCompletionStatusEvent would be a handler in case there is something wrong with the port so that the application does not endlessly loop trying to receive a fax when the port is not working. Below is one possible implementation of this, refer to the main sample for another.

C# Sample

```

// Form Level Variable
int    _receiveErrors;

private void _faxing_FaxCompletionStatus(object sender, FaxEventArgs args)
{

    // Other Completion Status Event Code
    ...

    // Check for bad device information to prevent endless cycling
    _receiveErrors = (DataTech.FaxManJr.FaxError.Error == args.Fax.FaxError) ?
_receiveErrors + 1 : 0;

    if (_waitingToSend)
    {
        // Let's send this fax that is ready to go out
        _waitingToSend = false;
        _faxing.Send(_fax);
    }
    else if (_receiveErrors > 10)
    {
        // device failed 10 times in a row
        System.Diagnostics.Debug.WriteLine("There are problems with your modem
Please check your modem and restart the application!");
    }
    else
    {
        // Ready to Receive another fax
        _faxing.Receive(1);
    }
}
}

```

In conclusion, when designing a desktop application users expect their application to be able to receive faxes as well as send them out. It is easy to add this functionality. The full sample automatically views the received faxes on successful reception as well.

1.6.10 Using the Debug Logging Feature

Sometimes issues may arise which require us to get additional information on whats happening with the FaxMan Jr control when its sending or receiving a fax. The **InitializeLogging** method of the **Faxing** class allows your application to capture the complete fax and modem communication data into a file which can be sent to our support technicians for study. This method should be called after instantiating the control before sending or receiving a fax.

The **Faxing** class also fires the **FaxMessage** event which provides the same data but allows your application to log it as it sees fit.

1.6.11 FaxMan Jr and Visual Studio 2005

When using FaxMan Jr with Visual Studio 2005 it is important to set the Target platform for solutions you build to x86 instead of the default AnyCPU. If you fail to set this option then applications using FaxMan Jr will not be able to run on 64 bit versions of Windows and will fail with a BadImageFormat exception when trying to load the FaxMan Jr. controls.

The reason for the exception is that the FaxMan Jr controls are written in Managed C++ code and compiled with Visual Studio 2003 and as such as 32 bit controls. Without the x86 Target option being set, the .Net framework will try to run the application as 64 bits and the 32 bit FaxMan Jr controls cannot be loaded. With the x86 option the application will be loaded as a 32 bit application and the controls can load successfully.

2 Printer Drivers

2.1 Using the FaxMan Printer Drivers

FaxMan includes printer drivers for Windows 95/98/ME, Windows NT 4, Windows XP, Windows 2000, Windows 2003 and Vista allowing applications running under those operating systems to create fax files just by printing.

Installing the Printer Drivers

The FaxMan printer drivers can be installed using the **Faxinstall.bat (Using Faxinstall.bat to install Printers.html)** file or by using the InstallPrinter method in either the ActiveX or .net components.

Your application must be running as a user with Administrative rights to be able to install the printer driver on XP, Windows 2000, Windows 2003 and Vista. Therefore you may want to consider installing the printer driver in your application's installer, on Vista this will generally already be running as Admin so your app can then run as a limited user.


How Printing Works

When an application prints to the FaxMan printer the driver will create a fax file in the directory specified by the printer properties or in the user's Temp directory if no directory is specified in the printer properties.

Once printing is complete, the FaxMan printer driver will look for a FaxMan application which has a Printer control whose PrinterName property matches the name of the printer. If an application is found the driver will fire the print control's PrintComplete event at which time the application can get the print job details. If no application is found the driver will attempt to start the application specified in the printer properties and when the application is loaded will fire the PrintComplete event. If no application is specified or it fails to load, the driver will write an entry to the Windows event log indicating a print job was generated but no application was found to notify.

2.2 Using the FaxInstall Utility

The faxinstall.bat file can be used to install a FaxMan Printer driver from a DOS console or from an Installer.

 The fmprint4.ocx file must be located in the same directory as the faxinstall.bat. The ocx file is required even if you are not using the Print OCX control but need not be registered if you are not using the control otherwise.

Usage

faxinstall.bat <Path To Your Fax Application> <Printer Name>

or

faxinstall.bat <Path to Config File>

By passing both the Application Path and Printer Name to the batch file the printer will be installed with the specified name and with a default set of printing properties. Using the Ini file method allows you to specify printing defaults for paper size, resolution and orientation as well as the name of the driver and the associated executable path.

The configuration ini file must be an ASCII text file in the following format and must have a .ini file extension:

```
[FaxInstall]
```

```
PrinterName=FaxTest Printer
```

```
AppPath=C:\Program Files\FaxMan3\Samples\Vbapp\Faxtest.exe
```

```
Resolution=2
```

```
PageSize=2
```

```
PageOrientation=1
```

The parameters which can be specified in the ini file include:

PrinterName

limit 24 characters

User friendly printer driver name

Default = "Fax Driver"

AppPath

limit 255 characters

full path name to application associated with driver

No Default

Resolution

1 = Low

2 = High, default

PageSize

1 = Letter, default

2 = A4

3 = Legal

PageOrientation

1 = Portrait, default

2 = Landscape

ReplaceFiles

0 = Replace All files

1 = Replace non-system files only, default

2 = Replace no files only copy if files not present

3 = Do not copy any files

PrintFilePath

limit 255 characters

full path name to directory for creating files

LogFile

limit 255 characters

Full path name of log file

2.3 Using the DeletePrn4 Utility

The DeletePrn4 utility is a command line application that can be used to remove FaxMan printers from a system. It supports removing both user-mode and kernel-mode printer FaxMan drivers and their associated files.

This utility can be called from an installer to remove prior instances of FaxMan drivers when upgrading to FaxMan Jr Version 2.

usage

deleteprn4 all

Deletes all FaxMan printers from a system and removes any associated printer driver files.

or

deleteprn4 <PrinterName>

Deletes the specified FaxMan printer from the system.

The DeletePrn4 utility is located in the *redist* subdirectory of the FaxMan install directory.

3 Appendices

3.1 Whats new in V 2.00

Version 2.00 of FaxMan Jr includes many new features including:

Class 2.1 Support for 33.6K Faxing

Class 2.1 modems include support for faxing at 33.6K which dramatically speeds up faxing when sending to or receiving from a 33.6K capable device. We recommend the Multitech ZBA5634 V.92 modems which are available in PCI, Serial and USB flavors and include Class 2.1 support. The Mainpine multi port modems that we resell also include several models that include class 2.1 support.

.Net Component

FaxMan Jr now includes fully managed .Net controls for faxing and handling the FaxMan printer driver. These controls are fully managed and do not rely on COM or need to be registered.

PDF Import Support

This optional new module allows FaxMan to create faxable files from an imported PDF file using the ImportFiles method or FaxCreate() function. Using this direct import support eliminates the troublesome process of automating Acrobat reader to print PDF files to the FaxMan printer driver. The FaxMan install includes a trial edition of this optional module which stamps imported PDF files with a watermark. For information or to order the add-on please visit our web site at <http://www.data-tech.com/products/fax/faxpdf.aspx>.

32 & 64 Bit User Mode Printer drivers for Windows 2000/XP/2003/Vista

FaxMan now includes 32 and 64 bit user mode printer driver which allows the drivers to be installed on Windows XP/2000/2003 and Vista operating systems.

3.2 Distributing the FaxMan ActiveX Components

The following 3 DLLs must be registered in the order that they appear below. You

can use Regsvr32.exe
to do this, if necessary.

ClassX.dll
ClassXps.dll
FMjr10.dll

These files are used for both importing and sending faxes and will need to be put in the same directory as the ClassX.dll.

IM32tif.dil
IM32bmp.dil
IM32pcx.dil
IM32fax.dil
IM32Xfax.del

The following file is only required if the optional PDF Import Add-on has been purchased:
xpdrasterizer.dll

Instructions for the FaxMan Jr. ActiveX Print Control

The following ocx will need to be registered.

fmprint4.ocx

All of these files should be installed in the System directory to ensure that they are available at application run time. All files have version resource information to allow your installation to be sure of installing the latest versions. The Printer Driver files can be installed using the **Faxinstall (Section 2.2)**.bat utility or via the FaxMan Jr Printer controls.

Windows 95 Installation Information

FaxMan Jr. is multithreaded and requires a version of the Microsoft COM libraries that support threaded objects. These libraries are part of Microsoft's DCOM installation.

DCOM is already installed on Windows 98,ME, NT4, Windows 2000/XP. If a client has installed Internet Explorer Version 4 or above on a Windows 95 machine it is also present there. Only Windows 95 machines that do not have Internet Explorer version 4 installed will not have DCOM installed. In this case you will need to install DCOM prior to installing FaxMan Jr.

One way to check if DCOM is installed from your installation script is to check in the registry.

Under: HKEY_LOCAL_MACHINE

Key: "SOFTWARE\Microsoft\OLE"

ValueName: "EnableDCOM"

Value: "Y" or "N"

Regardless of this value the presence of this ValueName indicates that the necessary files are installed.

If DCOM is not installed DCOM will need to be installed before the FaxMan Jr. components can be registered. So install DCOM98 using the following command line parameter to

run in silent mode:

/q:u

and if you want to handle the reboot from within your installation script use the following command line parameter to prevent reboot:

/r:n

If you want to continue your installation once the computer has rebooted add the following entry to the registry before you reboot.

Under: HKEY_LOCAL_MACHINE

Key: "SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce"

ValueName: "Your App"

Value: "c:\mypath\setup.exe"

Where "c:\mypath\setup.exe" points to your setup package.

3.3 Contacting DTI

3.3.1 FaxMan Jr. Technical Support

FaxMan Jr includes 60 days of free phone, email, fax and web support. The sixty day support period starts with your first support call, after that several plans are available to provide various levels of support. Visit our website at **www.data-tech.com** (<http://www.data-tech.com/>) for complete plan details and to order.

Before contacting DTI for technical support, be sure to thoroughly read and understand the FaxMan Jr. documentation, and check our online knowledge base for more up to date information about FaxMan Jr.

Our website also hosts our Online Update Center, which allows you to obtain updates to the latest FaxMan Jr. release. You'll need to have your serial number handy and have registered your product to be able to use this service. You can also sign up for the FaxMan mailing list on our website. Once signed up, you'll receive automatic e-mail notifications of updates and other important information about FaxMan Jr.

For communication problems with FaxMan Jr. you should also consult the **Common Problems (Common_Problems.html)** section

Be sure to check out the **DTI Knowledge base** (<http://www.data-tech.com/support/newkb.aspx>) located on our website for answers to common issues. Our website also contains our **Online Update center** (<http://www.data-tech.com/myproducts/login.aspx>) (http://www.data-tech.com/Scripts/online_update_center.asp) which you can use to download updates to the FaxMan Jr. product. You'll need your serial number, zipcode and country to be able to download updates. Be sure to send in your registration information so you can take advantage of this service.

Data Techniques, Inc.

PO Box 606

Burnsville, NC 28714

Email: **support@data-tech.com**

Web: **www.data-tech.com** (<http://www.data-tech.com/>)

Tech Support: 828-682-0161 9am to 5pm EST

Sales 828-682-4111

Fax: 828-682-0025

3.4 Converting FMF Files to TIFF Format

Writing FMF Files using ImageMan

When writing FMF files using ImageMan, it's important to make sure the files follow these guidelines:

- The Files must be 1728 pixels wide
- The Vertical resolution must be 98 (Low Res) or 196 (High Res). This is specified using the TIFF_YRES option block setting.
- The image data must be 1 Bit (Black & White)
- Compression = Group 3
- Fillorder=2 (reverse byte order)

3.5 Class 2/2.0 Modem Hangup Codes

Class 2/2.0 Modem Hangup Codes

Code Description

0-9 Call Placement and Termination

0 Normal and proper end of connection

1 Ring detected without successful handshake

2 Call aborted from either +FK or AN

3 No Loop Current

10-19 Transmit Phase A & Misc. Errors

10 Unspecified Phase A error

11 No Answer (T.30 T1 timeout)

20-39 Transmit Phase B Hangup Codes

- 20 Unspecified Transmit Phase B error
- 21 Remote cannot receive or send
- 22 COMREC error in transmit Phase B
- 23 COMREC invalid command received
- 24 RSPREC error
- 25 DCS sent three times without response
- 26 DIS/DTC received 3 times; DCS not recognized
- 27 Failure to train at 2400 bit/s or +FMS value
- 28 RSPREC invalid response received

40-49 Transmit Phase C Hangup Codes

- 40 Unspecified Phase C Transmit error
- 43 DTE to DCE data underflow

50-69 Transmit Phase D Hangup Codes

- 50 Unspecified Phase D transmit error
- 51 RSPREC error
- 52 No response to MPS repeated 3 times
- 53 Invalid response to MPS
- 54 No response to EOP repeated 3 times
- 55 Invalid response to EOM
- 56 No response to EOM repeated 3 times
- 57 Invalid response to EOM
- 58 Unable to continue after PIN or PIP

70-89 Receive Phase B Hangup Codes

- 70 Unspecified Phase B receive errors
- 71 RSPREC error
- 72 COMREC error
- 73 T.30 T2 timeout, expected page not received
- 74 T.30 T1 timeout, after EOM received

90-99 Receive Phase C Hangup Codes

- 90 Unspecified Phase C receive errors

91 Missing EOL after 5 seconds
 92 Unused Code
 93 DCE to DTE buffer overflow
 94 Bad CRC or frame (ECM or BFT modes)

100-119 Receive Phase D Hangup Codes
 100 Unspecified Phase D receive errors
 101 RSPREC invalid response received
 102 COMREC invalid response received
 103 Unable to continue after PIN or PIP

120-255 Reserved Codes

3.6 Modem Configuration Information

Modem Configuration Information

FaxMan Jr. uses a default modem initialization string that works with most fax modems.

In order for the FaxMan Jr. control to work correctly with the modem the modems needs to have

flow control enabled. The flow control can be set to hardware flow control, software flow

control or both. Certain modems, in particular some USR models, do not have flow control

enabled by default.

Some modems will need special initialization strings. Though these will not be appropriate for

every model under each listed manufacturer, here is a list of init strings that may be helpful:

FaxMan Jr. Default Init String:

```
AT&FE0V1&C1&D2S0=0
```

Digi Acceleport:

AT&FE0V1&K6&C1&D2S0=0

US Robotics Sporsters:

AT&F1E0V1&C1&D2S0=0

alternate USR string to enable both hardware and software flow control.

AT&F1&H3&I2&R2E0V1&C1&D2S0=0

Practical Peripherals:

AT&FE0V1&C1&D2&K4S0=0

MultiTech:

AT&FE0V1&C1&D2&E5S0=0

or for both Hardware & Software flow control:

AT&FE0V1&C1&D2&E4&E11S0=0

Hayes:

AT&FE0V1&C1&D2/Q1S0=0

To set these initialization strings you can set the **DeviceInit (DeviceInit_Property.html)** string for any of the com ports you need to modify this for.

See the **DeviceInit (DeviceInit_Property.html)** property.

3.7 Coverpages

Coverpages

FaxMan Jr. supports two types of coverpages, the FaxMan Jr. coverpage format and also Microsoft Fax coverpages. By using the Microsoft Fax Coverpage editor that shipped with Windows 95, it is possible to create graphically rich coverpages that can be sent using FaxMan Jr..

FaxMan Jr. currently ships with two standard Coverpages. The pages are:

Fax

DATE

TIME

PAGES

TO

COMPANY

FAX NUMBER

NOTE

FROM

COMPANY

VOICE NUMBER

FAX NUMBER

Cover1.pg

Requires FaxMan.fmf file (used as background bitmap)

Facsimile Cover Sheet

To:
Company:
Fax:

From:
Company:
Phone:
Fax:

Date:

Pages including this
cover page:

Comments

Cover2.pg

The following sample Microsoft Fax Coverpages have been installed in your windows system directory:

Generic.cpe

Urgent!.cpe

Confidential!.cpe

For Your Information.cpe

If you are interested in creating your own FaxMan Jr. format coverpages you may do so by creating a cover page file using the information that follows within this Appendix.

3.8 Banner And Coverpage Formatting Characters

Banner And Coverpage Formatting Characters

Banners:

A banner string may consist of any of three parts : a left-justified portion, a center-justified portion, and a right-justified portion. Each part is separated by a vertical bar (`|`). The banner string may be of any length, however, only one line of text will appear at the top of a fax. You may insert format characters into the banner string in order to include some information concerning the fax, see the description of format characters below. If the fax banner is empty then no fax banner will be sent.

The banner string has a limit of 64k, a great deal more than would fit across a single page, and we do not limit the text in any given justification area. Therefore there is the potential that the center-justified portion could overwrite the left-justified portion and the right-justified portion could overwrite the both the left and center-justified portions. The user would not normally be aware of this situation because the banner is printed only on out going faxes.

Format Characters:

The following strings are valid formatting characters for both the Coverpage Comments and the Banner string:

Item	Description	Property Name
%d	date (dd-mmm-yyyy format, e.g. 03-Feb-1995)	
%t	send time (24-hour format, e.g. 6:00pm = 18:00:00)	
%p	total # of pages in the fax	
%c	current page	
%r	Recipient Name	FaxName
%y	Recipient Company	FaxCompany
%s	Sender's Name	UserName
%m	Sender's Company	UserCompany
%i	Sender's Fax ID	LocalId

%u	Subject of fax	FaxSubject
%f	Fax Number	FaxNumber
%o	Comments	FaxComments
%x	Senders Fax #	UserFaxNumber
%h	Senders Voice #	UserVoice

3.9 Visual Basic Source Code for Creating a Cover Page

Visual Basic Source Code for Creating a Cover Page

This sample is a VB 4.0 .FRM file which contains the code to write a simple coverpage.

```

VERSION 4.00
Begin VB.Form frmMain
Caption = "Form1"
ClientHeight = 8430
ClientLeft = 3225
ClientTop = 2250
ClientWidth = 6690
Height = 8835
Left = 3165
LinkTopic = "Form1"
ScaleHeight = 8430
ScaleWidth = 6690
Top = 1905
Width = 6810
End
Attribute VB_Name = "frmMain"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Dim fhndl As Integer

```

```
Private Type FontObj
  OpCode As Integer
  RecLen As Integer
  Name As String * 20
  Size As Integer
  Bold As Integer
  Italic As Integer
  Underline As Integer
  Color As Long
End Type
```

```
Private Type TextObj
  OpCode As Integer
  RecLen As Integer
  X1 As Integer
  Y1 As Integer
  X2 As Integer
  Y2 As Integer
  Color As Long
  Flags As Integer
End Type
```

```
Private Type LineObj
  OpCode As Integer
  RecLen As Integer
  X1 As Integer
  Y1 As Integer
  X2 As Integer
  Y2 As Integer
  Width As Integer
  Style As Integer
  Color As Long
End Type
```

```
Private Type BitMapObj
```

```
OpCode As Integer
```

```
RecLen As Integer
```

```
X1 As Integer
```

```
Y1 As Integer
```

```
X2 As Integer
```

```
Y2 As Integer
```

```
End Type
```

```
Private Sub Form_Load()
```

```
Dim FileName As String, TextStr As String
```

```
Dim BitMapRec As BitMapObj
```

```
Dim BitMapName As String
```

```
fhndl = FreeFile
```

```
Open "d:\faxman2\vbsample.pg" For Binary Access Write As fhndl
```

```
'Add a bitmap called cologo.fmf to the coverpage. The logo file
```

```
'must reside in the FaxMan Jr. server directory along with the cover pg file
```

```
BitMapName = "cologo.fmf" + Chr(0)
```

```
BitMapRec.OpCode = 1
```

```
BitMapRec.RecLen = Len(BitMapRec) + Len(BitMapName)
```

```
Put fhndl, , BitMapRec
```

```
Put fhndl, , BitMapName
```

```
DrawBox 6440, 3400, 11700, 4400
```

```
DrawBox 480, 4720, 5900, 7500
```

```
DrawBox 6440, 4720, 11700, 7500
```

```
DrawBox 480, 7750, 11700, 12500
```

```
DrawLine 480, 8500, 11700, 8500
```

SetFont "Arial Black", 450, 1

WriteText 600, 2400, 7800, 3000, "Sample FaxMan Jr. Cover Page"

SetFont "Times New Roman", 900, 1

WriteText 600, 3360, 2640, 4080, "FAX"

SetFont "Times New Roman", 240, 0

WriteText 7000, 3650, 10400, 4080, "Date:"

WriteText 7000, 3970, 10080, 4280, "Number of pages including cover sheet:"

WriteText 1000, 5500, 2500, 5900, "Name:"

WriteText 1000, 6000, 2500, 6400, "Company:"

WriteText 1000, 7000, 2500, 7400, "Fax phone:"

WriteText 7000, 5500, 10000, 5900, "Name:"

WriteText 7000, 6000, 10000, 6400, "Company:"

WriteText 7000, 6500, 10000, 6900, "Phone:"

WriteText 7000, 7000, 10000, 7400, "Fax phone:"

WriteText 2050, 8050, 11600, 8500, "%u"

WriteText 1000, 9000, 11600, 12400, "%o"

SetFont "Times New Roman", 300, 1

WriteText 7950, 3600, 11600, 4080, "%d"

WriteText 10320, 3920, 11600, 4280, "%p"

WriteText 750, 5000, 2500, 5400, "To:"

WriteText 6750, 5000, 10000, 5400, "From:"

WriteText 2050, 5450, 5800, 5900, "%r"

WriteText 2050, 5950, 5800, 6400, "%y"

WriteText 2050, 6950, 5800, 7400, "%f"

WriteText 8050, 5450, 11600, 5900, "%s"

```
WriteText 8050, 5950, 11600, 6400, "%m"
```

```
WriteText 8050, 6450, 11600, 6900, "%h"
```

```
WriteText 8050, 6950, 11600, 7400, "%x"
```

```
WriteText 750, 8000, 2500, 8400, "Subject:"
```

```
WriteText 750, 8700, 2500, 9100, "Comments:"
```

```
Close fhndl
```

```
End
```

```
End Sub
```

```
Public Sub WriteText(X1 As Integer, Y1 As Integer, X2 As Integer, Y2 As Integer,  
TextStr As String)
```

```
Dim TextRec As TextObj
```

```
TextRec.OpCode = 5
```

```
TextRec.X1 = X1
```

```
TextRec.Y1 = Y1
```

```
TextRec.X2 = X2
```

```
TextRec.Y2 = Y2
```

```
TextStr = TextStr + Chr$(0)
```

```
TextRec.RecLen = Len(TextRec) + Len(TextStr)
```

```
Put fhndl, , TextRec
```

```
Put fhndl, , TextStr
```

```
End Sub
```

```
Public Sub SetFont(Name As String, Size As Integer, Bold As Integer)
```

```
Dim FontRec As FontObj
```

```
FontRec.OpCode = 2
```

```
FontRec.RecLen = Len(FontRec)
```

```
FontRec.Name = Name + String$(20 - Len(Name), Chr$(0))
```

```
FontRec.Size = Size
```

```
FontRec.Bold = Bold
```

```
Put fhndl, , FontRec
```

```
End Sub
```

```
Public Sub DrawLine(X1 As Integer, Y1 As Integer, X2 As Integer, Y2 As Integer)
```

```
Dim LineRec As LineObj
```

```
LineRec.OpCode = 4
```

```
LineRec.RecLen = Len(LineRec)
```

```
LineRec.X1 = X1
```

```
LineRec.Y1 = Y1
```

```
LineRec.X2 = X2
```

```
LineRec.Y2 = Y2
```

```
LineRec.Color = 0
```

```
LineRec.Width = 10
```

```
Put fhndl, , LineRec
```

```
End Sub
```

```
Public Sub DrawBox(X1 As Integer, Y1 As Integer, X2 As Integer, Y2 As Integer)
```

```
DrawLine X1, Y1, X2, Y1
```

```
DrawLine X1, Y2, X2, Y2
```

```
DrawLine X1, Y1, X1, Y2
```

```
DrawLine X2, Y1, X2, Y2
```

```
End Sub
```

3.10 C Source Code for Creating a Cover Page

C Source Code for Creating a Cover Page

```
#include <windows.h>
```

```
#include <sys/stat.h>
```

```
#include <io.h>
```

```
#include "portable.h"
```

```
#include "cover.h"
```

```
#define STRING( x, y, x2, y2, s, clr, flg ) \
```

```
tx.nX1 = x; tx.nY1 = y; tx.nX2 = x2; tx.nY2 = y2; tx.rgbColor = clr; tx.nFlags = flg; \
```

```
Rec.nOpCode = TEXT_OBJ; Rec.nRecLen = sizeof(Rec) + sizeof(tx) + sizeof(Str) + 1; write( fd, &Rec, sizeof(Rec) ); \
```

```
write( fd, &tx, sizeof(tx)); write( fd, Str, sizeof(Str)+1);
```

```
main()
```

```
{
```

```
int fd;
```

```
COVERREC Rec;
```

```
FONTREC fr;
```

```
TEXTREC tx;
```

```
LINEREC lr;
```

```
BITMAPREC bm;
```

```
char Str[100];
```

```
fd = _creat("cover.pg", _S_IWRITE );

if( fd < 0 ) {
    printf("Error creating file....");
    exit(-1);
}

// Write our background record
Rec.nOpCode = BITMAP_OBJ;
strcpy( Str, "FaxMan.fmf");
Rec.nRecLen = sizeof(Rec) + sizeof(bm) + strlen(Str) + 1;

write( fd, &Rec, sizeof(Rec) );

write( fd, &bm, sizeof(bm) );

write( fd, Str, strlen(Str)+1 );

// Write a Font Record
Rec.nOpCode = FONT_OBJ;

strcpy( fr.szFontName, "Arial" );
fr.nSize = 300;
fr.bBold = 0;
fr.bUnderline = 0;
fr.bItalic = 0;
fr.rgbColor = RGB(0, 0, 0);

write( fd, &Rec, sizeof(Rec) );
write( fd, &fr, sizeof(fr) );

// Write a Text record
strcpy( Str, "%s");
STRING( 720, 10000, 3500, 11400, Str, 0, DT_RIGHT );
```

```

// Write Text
strcpy( Str, "%d");
STRING( 7200, 1620, 10800, 2000, Str, 0, DT_LEFT );

strcpy( Str, "%t");
STRING( 7200, 2500, 10800, 3000, Str, 0, DT_LEFT );

strcpy( Str, "%p");
STRING( 7200, 3400, 10800, 7000, Str, 0, DT_LEFT );

strcpy( Str, "%r");
STRING( 7200, 5000, 10800, 7000, Str, 0, DT_LEFT );

strcpy( Str, "%f" );
STRING( 7800, 6650, 10800, 14400, Str, 0, DT_LEFT );

strcpy( Str, "%o" );
STRING( 6048, 7700, 10800, 14400, Str, 0, DT_LEFT | DT_WORDBREAK );

close(fd );
}

```

3.11 Cover Page Reference

Cover Page Reference

The FaxMan Jr. server supports both its own cover page format and also the Microsoft Fax Coverpage format. Using the Microsoft Fax Coverpage Editor which shipped with Windows 95, it is possible to create graphically rich coverpages that can be used with FaxMan Jr.. Now that FaxMan Jr. includes this support we would recommend that developers use these coverpages instead of the FaxMan Jr. formatted files.

Note: to acquire the Microsoft Fax Editor from the Win98 installation CD, see the following DTI Knowledgebase article

<http://www.data-tech.com/content/kbarticle.aspx?ID=82>

The FaxMan Jr. coverpage form is documented in this appendix if you choose to use it instead of the MS Fax format.

The FaxMan Jr. coverpage format isn't too difficult to create, as you'll see – if you're familiar with files at all, you shouldn't have any trouble.

The cover page file format is a simple binary format consisting of one or more variable length records representing objects such as fonts, text, lines, rectangles and Bitmaps. These files can be created from any language that supports writing binary files. When using C or C++ to create cover pages be sure to open/create the file in binary mode so that CR/LF translation will be disabled.

The current cover page format has no header structure and begins with the first object in the file. Future releases will probably include a header.

Object Types

Font

The Font object defines a font to be used for subsequent Text output. This record must be output before any Text records.

The structure is:

SHORT nOpCode; Always set to 0x02 for Font Object

SHORT nRecLen; Set to size of entire structure

char szFontName[20]; Name of Windows Typeface

SHORT nSize; Size in TWIPS (1/1440")

WORD bBold; True if Bold

WORD bItalic; True if Italic

WORD bUnderline; True if Underlined

COLORREF rgbColor; Color of Font

Text

The Text object defines the placement, formatting and text to be printed. Multiple text objects may be used.

SHORT nOpCode; Always set to 0x05 for Text Object

SHORT nX2; X coordinate of bottom right corner of Bounding box

SHORT nY2; Y coordinate of bottom right corner of Bounding Box

COLORREF rgbColor; This is not used.

SHORT nFlags; Flags which specify text formatting. Same flags as DrawText.

Char text[1] Random length text data.

nFlags specifies an array of flags that determine how to draw the text. This parameter can be a combination of the following values:

Value Meaning

DT_BOTTOM (0x08) Specifies bottom-aligned text. This value must be combined with DT_SINGLELINE.

DT_CENTER (0x01) Centers text horizontally.

DT_EXPANDTABS (0x40) Expands tab characters. The default number of characters per tab is eight.

DT_LEFT (0x00) Left-aligns text.

DT_NOPREFIX (0x800) Turns off processing of prefix characters. Normally, DrawText interprets the mnemonic & as a directive to underscore the character that follows, and the mnemonic && as a directive to print a single &. By specifying DT_NOPREFIX, this processing is turned off.

DT_RIGHT (0x02) Right-aligns text.

DT_SINGLELINE (0x20) Specifies single line only. Carriage returns and linefeeds do not break the line.

DT_TOP (0x00) Specifies top-aligned text (single line only).

DT_VCENTER (0x04) Specifies vertically centered text (single line only).

DT_WORDBREAK (0x10) Specifies word breaking. Lines are automatically broken between words if a word would extend past the edge of the rectangle specified by the lprc parameter. A carriage return–linefeed sequence will also break the line.

Note that the DT_CALCRECT, DT_EXTERNALLEADING, DT_INTERNAL, DT_NOCLIP, and DT_NOPREFIX values cannot be used with the DT_TABSTOP value.

Line

The line object causes a line of the specified size and color to be drawn.

SHORT nOpCode; Always set to 0x04 for Line Object

SHORT nRecLen; Set to size of entire structure

SHORT nX1; X coordinate of starting point

SHORT nY1; Y coordinate of starting point

SHORT nX2; X coordinate of end point

SHORT nY2; Y coordinate of end point

SHORT nWidth; Width of line in TWIPS

SHORT nStyle; Style of Line, same as PENSTYLE in CreatePen

COLORREF rgbColor; Color of line

Rectangle

Draws a rectangle at the specified location. The rectangle can be filled or hollow.

SHORT nOpCode; Always set to 0x03 for Rectangle Object

SHORT nRecLen; Set to size of entire structure

SHORT nX1; X coordinate of Upper Left corner of rectangle

SHORT nY1; Y coordinate of Upper Left corner of rectangle

SHORT nX2; X coordinate of bottom right corner of rectangle

SHORT nY2; Y coordinate of bottom right corner of rectangle

COLORREF rgbColor; Color of rectangle

WORD bFilled; True if filled

COLORREF rgbFillColor; Fill Color

SHORT nPenWidth; Width of Pen in TWIPS

SHORT nPenStyle; Style of Line, same as PENSTYLE in CreatePen

Bitmap

Adds a bitmap to the page. Currently this bitmap must be in FaxMan Jr. FMF format, Black & white, and 1728 pixels wide. The vertical resolution should be 200 DPI. Currently only 1 bitmap may be placed per page and the bitmap will be placed at the top of the page as the X1,Y1,X2,Y2 coordinates are ignored.

SHORT nOpCode; Always set to 0x01 for Bitmap Object

SHORT nRecLen; Set to size of entire structure

SHORT nX1;

SHORT nY1;

SHORT nX2;

SHORT nY2;

char filename[1]; Name of FMF file (Without path information)

3.12 Error Codes

Error Codes

Errors that can be raised by SendFax:

FAXJR_ERR_CLASS

FAXJR_ERR_PORT

FAXJR_ERR_NO_FILELIST

FAXJR_ERR_PORT_IN_USE

FAXJR_ERR_FAXDEVICE

All Custom Control Error Codes:

```

FAXJR_ERR_NO_FILELIST = 32000 //problem with filelist when sending
FAXJR_ERR_PORT = 32001 //bad port #
FAXJR_ERR_CLASS = 32002 //bad fax class setting
FAXJR_ERR_PORT_IN_USE = 32003 //port is already in use
FAXJR_ERR_FAXDEVICE = 32004 //unable to create FAXDEVICE
                             (ClassX.dll , ClassXps.dll, or
                             FMjr10.dll may not be registered)
FAXJR_ERR_NO_DETECT_OBJECT = //unable to create auto-detect object
32005 (ClassX.dll may not be registered)
FAXJR_ERR_BAD-RINGCOUNT = 32006
FAXJR_ERR_NO_THREAD = 32007

```

3.13 FaxmanJr License

FaxManJr License Agreement

1. IMPORTANT NOTICE

This is a legal agreement between Data Techniques and you, the licensee. BY OPENING THIS SEALED DISK PACKAGE, YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, RETURN THE UNOPENED DISK PACKAGE TO THE PLACE WHERE YOU OBTAINED IT WITHIN 60 DAYS FROM THE DATE OF SHIPMENT FOR A FULL REFUND.

2. ON ONE COMPUTER

Data Techniques licenses this product for use on a single CPU only. You must obtain a licensed copy (or special license) for each programmer or workstation on a network on which the product is used, whether in source or object format.

3. LICENSE TO REPRODUCE, USE AND DISTRIBUTE

You may distribute only the FaxManJr files listed as Redistributable in the Online Help and Documentation with your end-user applications. You may not distribute any FaxManJr files not listed as Redistributable.

You may only use the FaxManJr software in end-user software that does not provide the end-user any development capability or ability to further redistribute the licensed code. You also may not use the FaxManJr code in a product which is competitive with the FaxManJr product including but not limited to a developer toolkit or similar product.

4. RUNTIME DISTRIBUTIONS

A Runtime License is required for each disposition of the product containing the Licensed Software.

You are granted 10,000 Runtime Licenses per year with the initial purchase of the Licensed Software. Additional Runtime Licenses must be purchased in advance of distribution.

5. AUDITS

You shall keep adequate records to accurately determine the payments due under this Agreement.

DTI shall have the right to have an independent certified public accountant inspect the relevant records on seven days written notice to verify the accuracy of such records. The Audit shall be at the expense of DTI, except where greater than 10,000 unlicensed Dispositions of the software are found, and in such cases the entire cost of such an audit shall be borne by you.

6. OWNERSHIP, COPYRIGHT AND TRANSFER

All copies of FaxManJr are owned by Data Techniques, Inc. or its suppliers and are protected by United States copyright laws and international treaty provisions. You must treat Data Techniques, Inc. software products as any other copyrighted material (e.g., book or musical recording), except that you may make a reasonable number of backups for archival purposes, provided that you have only one working copy at a time for each copy of the product licensed to you. You may not transfer, sublicense, rent, lease or assign any of the above license or any Data Techniques, Inc. software.

7. TERM AND TERMINATION

You may terminate the above license at any time by returning or destroying all copies of FaxManJr in your possession and notifying Data Techniques, Inc. The above license will terminate immediately if you infringe upon Data Techniques, Inc.'s copyrights or breach this agreement.

8. LIMITED WARRANTY

Data Techniques provides a 60-day money-back guarantee for the FaxManJr product when purchased directly from Data Techniques. You may return the version of FaxManJr at any time within 60 days of the date of purchase for any reason whatsoever by calling Data Techniques, Inc. and obtaining a return authorization number (RMA). Under no circumstances will Data Techniques, Inc. accept a returned product without an RMA, or a package in which the source diskette envelope has been opened.

9. NO LIABILITY FOR CONSEQUENTIAL DAMAGES

IN NO EVENT SHALL DATA TECHNIQUES, INC. OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER STEMMING FROM THE USE OR MISUSE OF THIS PRODUCT, EVEN IF DATA TECHNIQUES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES,

THE ABOVE LIMITATION MAY NOT APPLY TO YOU.