equisys
DocumentDynamics

Zetafax 2012 API Help

equisys ZETAfax®
Point. Click. Fax.

Fax software solutions for business

Zetafax 2012 API Help

Printed: 2012

Table of Contents

The API - Getting Started

Introduction to the API

Without the API

The normal method of submitting messages using Zetafax is to use the client program to manually specify the recipients and message settings. When sending a fax, when a file is printed to the Zetafax printer, or a file is selected for sending from within the client program, the addressing screens are automatically displayed and filled in by the user.

Using the API

Using the Zetafax API, there are five additional ways of submitting message to the Zetafax server. These additional methods have been designed to the automatic sending of messages from other applications not only feasible, but extremely easy to implement.

SMS meassages as well as fax messages can be submitted to the Zetafax server with the Zetafax API. The ZSUBMIT program makes it simple to send SMS messages directly from other applications to mobile phones.

Submit files

The simplest method of sending messages is to create an ASCII file in a directory on a fileserver - a "SUBMIT" file. This file can contain both the contents of the message and the addressing information, or these may be broken into two files. Zetafax scans the directory for new files and automatically handles any files it finds, sending messages to either a fax addressee or mobile phone via SMS. A machine with the Zetafax client software installed can be used to monitor the process and resubmit any faxes that fail repeatedly.

Details on how to create SUBMIT files are given in Creating SUBMIT files.

Embedded addressing

Addressing instructions can be embedded into documents created by word processors and other applications to indicate where a fax should be sent, together with a wide range of settings such as the time of sending, priority, resolution, coversheet, and letterhead to use. The commands are embedded in the document using a special syntax; for example:

```
%%[Fax:123 456 7890]
```

When the document is printed using the Zetafax printer driver, the embedded addressing commands are used by the Zetafax client program in place of the addressing dialogs. Provided adequate details are given using embedded commands, no dialogs will be displayed and the fax or faxes will be submitted to the Zetafax server automatically.

When sending faxes automatically from other Windows applications, it may be simplest to embed the addressing information in the message itself using embedded commands, prior to printing it using the Zetafax Windows printer driver. This is supported as one of the API toolkit alternatives.

DDE commands

Dynamic Data Exchange (DDE) may be used to pass addressing information across from a Windows application to the Zetafax client, prior to printing a message using the Zetafax Windows printer driver.

COM and C Language libraries

The most powerful way of passing faxes across to the Zetafax server for sending is using the COM or C language API. This allows for the monitoring of the status of a queued message, including the retrieval of a full history log once a message has completed.

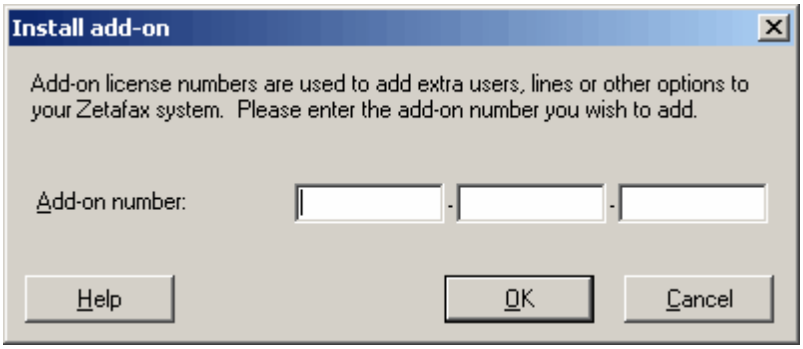
Installing the API

The Zetafax API offers five methods of automatically submitting messages to the Zetafax server, the automatic submission program, embedded addressing, DDE and the COM and C language libraries which are all independent. This section explains how each feature is installed on your system. You should refer to the appropriate section once you have decided which method you intend to use.

Licensing of the API depends on which API product you have purchased. For users of Zetafax network fax software, please refer to the section entitled "Zetafax API Toolkit", and for Equisys (Independent Software Vendor) ISV partners, please refer to section entitled "Zetafax engine".

Zetafax API Toolkit

If Zetafax is currently up and running on your network, then the automatic submission, embedded addressing and DDE features of the Zetafax API are already installed! You only need to activate these features by adding the API toolkit license number in the modify license details dialog in the **Zetafax Configuration** program.



Fax engine

If Zetafax has been supplied to you as an Equisys (Independent Software Vendor) ISV partner, there is no need to add a separate API license as described above. The Zetafax fax engine products supplied to ISV partners include the API license as standard within the starter system license number.

ZSUBMIT

The ZSUBMIT program is distributed with the Zetafax server programs, and is stored in the same directory as the other programs (zfax \SERVER by default, where zfax is the server base directory specified when Zetafax was installed).

Embedded addressing

Embedded addressing requires the Zetafax printer driver to be installed on all of the Zetafax clients wishing to use this feature. The Zetafax printer driver is installed by running the Workstation Setup program (WKSETUP) - refer to the Installation and Configuration Guide for detailed instructions on using this program. There is limited support for embedded addressing available without the API, and the full functionality is provided when the API toolkit has been purchased.

DDE

Support for Dynamic Data Exchange (DDE) is built in to the Zetafax client software. As for embedded addressing, there is limited support for DDE in the standard Zetafax software. Full support is available only when the API toolkit has been purchased.

C language API

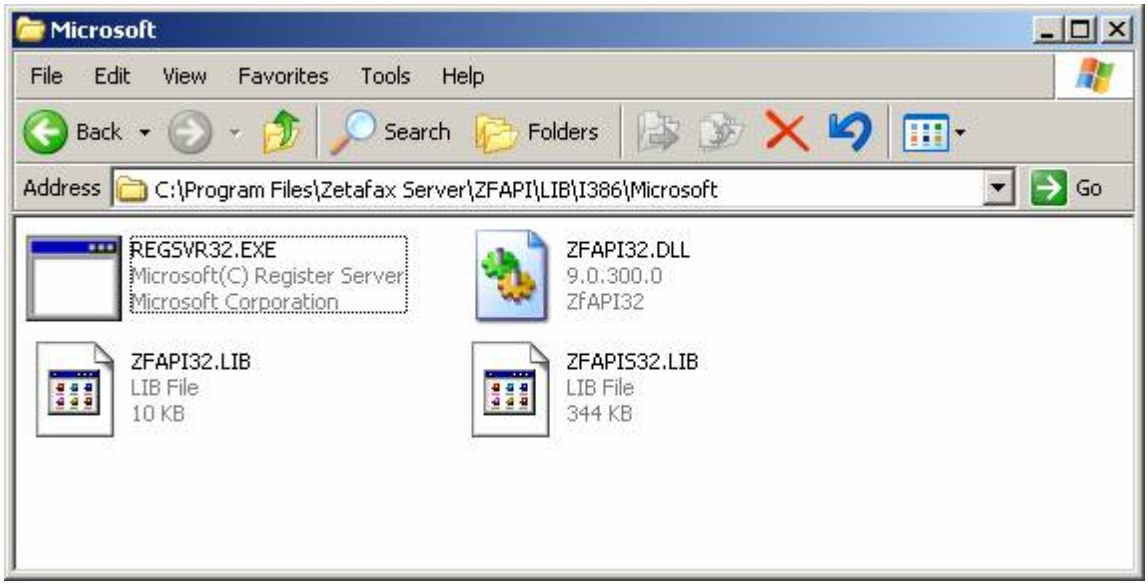
The C language libraries and 32-bit dynamic link library are provided separately in a self-extracting executable. The appropriate libraries need to be copied into your development environment and if used,

the DLL needs to be copied to the relevant location. Please refer to C language API for more detailed information.
Programs written using the Zetafax C language API may only be run on Zetafax systems with an API license.

COM API

Like all COM components ZfAPI32.dll must be registered before it can be used. This is done using the program regsvr32 as follows:

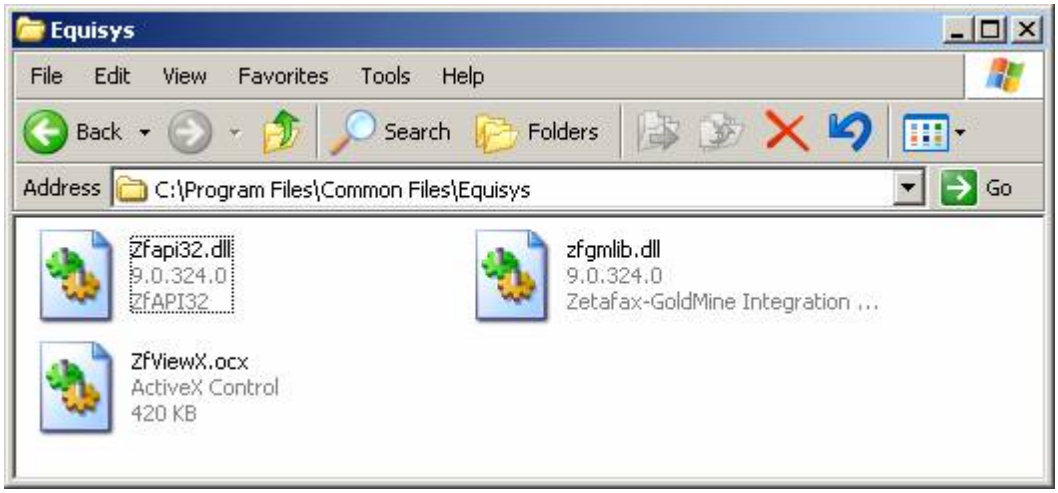
1. Ensure the program regsvr32 is present in the same location as the ZFAPI32.dll file.



2. In a DOS window, set to this directory, use the following command to run the registration program: Run %system32%\regsvr32 ZfAPI32.dll.

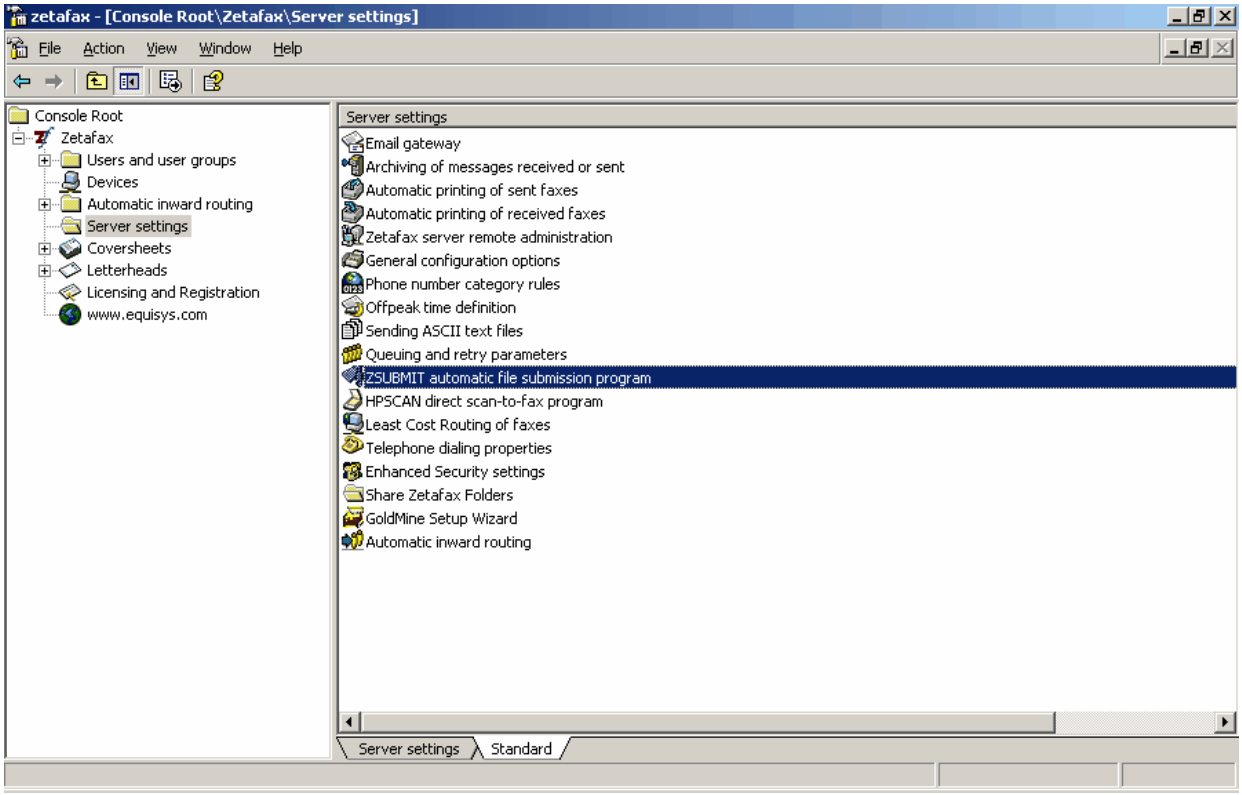
3. A dialog box should appear confirming that the registration was successful.

Note: If you have installed the Zetafax client onto your development computer, the COM API is installed and registered automatically. It is installed by default to C:\Program Files\Common Files\Equisys.

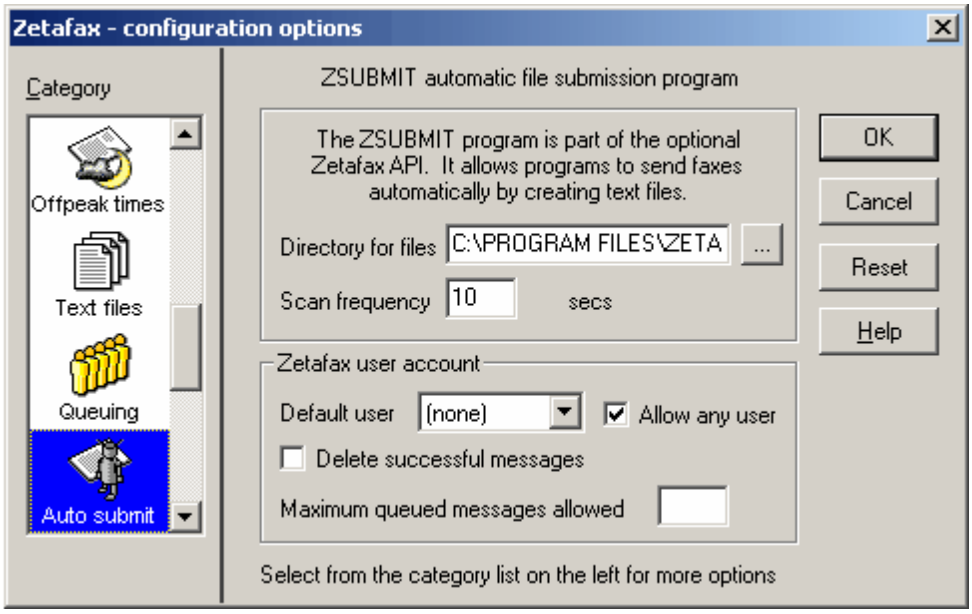


API configuration

From the Zetafax Configuration program, located on the Zetafax server, you can adjust the parameters specific to the ZSUBMIT program. Select the ZSUBMIT automatic file submission program option from the Server settings options:



Selecting this option will open the Zetafax - configuration options dialogue:



These options are already setup with values, allowing the ZSUBMIT program to be used however, you can change these settings to better meet your needs:

Directory for files

Specify the directory which the ZSUBMIT program looks in for SUBMIT files to send. Any files found whose names match the given specification (normally "*.SUB" - see Expert setup options will be interpreted as SUBMIT format files and submitted to the server for sending.

Scan frequency

Specify the polling interval, in seconds. This is how often the program looks for SUBMIT files in the directory for sending and check the status of messages already queued. Reducing this value will increase the processing overhead of the program.

Zetafax user account

Set up the user account details for use with auto submission of faxes.

Default user

Specify the Zetafax user account to be used when submitting files. All messages will be sent as if by this user, unless a user name is specified in the SUBMIT file. If the user name is left blank then each SUBMIT file must specify a user name.

Allow any user

Specify whether users can specify the Zetafax user name in the SUBMIT file, overriding the name given in the Default user field. Do not check this box if you want to ensure that messages can only be submitted as the given user when using the ZSUBMIT program (e.g. for permissions or security reasons).

Delete successful messages

Check this box if faxes that have been sent successfully (i.e. faxes that would be displayed on the Zetafax Client with a tick) should be deleted automatically. This is useful in systems where a large number of faxes are being submitted automatically and you just want to check for any which have failed.

Maximum queued messages allowed

Specify the maximum number of messages that may be active in the queue for this user at any one time.

If the value is 0 then any SUBMIT file found in the directory will be submitted to the server immediately, when it is found. Otherwise the file is only submitted if the number of active messages is less than the specified value. The program will wait for one of the existing messages to complete, before submitting the file.

OK

Save any changes that have been made to the ZSUBMIT automatic file submission program configuration, and exit the Zetafax - configuration options dialog.

Cancel

Do not save changes made to the ZSUBMIT automatic file submission program configuration, and exit the Zetafax - configuration options dialog.

Reset

Reset the ZSUBMIT automatic file submission program configuration to its default settings. The settings are not saved until either OK is clicked or another category icon is selected from the scroll (left).

If you use the scrolling icons on the left to move to another Category, any changes made to the ZSUBMIT automatic file submission program configuration are saved.

Related topics
The ZSUBMIT program

How to Develop and distribute your fax-enabled application

Developing your application

1) Request an Equisys API Developer Kit

The Equisys API Developer Kit is available to all Equisys ISV partners. Included in the kit is a 5 user, 2 lines Zetafax server and the developer tools required to develop fully integrated applications.

Full documentation on how to install the server software and on using the development tools is included in the documentation supplied with the developer kit. Additional resources are available free of charge in the support section of the Equisys web site http://www.equisys.com/

2) Install the Zetafax API and developer tools

The Zetafax API runs under Windows, and supports the same operating systems as the Zetafax server. One or more fax modems, active ISDN controllers or intelligent fax boards are required for sending and receiving faxes. Installation is a very simple process and should take less than 10 minutes.

During the install process, you will be required to enter a license number supplied with the system, which enables both the server and development tools.

You will also need to create a Zetafax user account to use for sending and receiving messages. This can be done during installation or afterwards, using the Zetafax Configuration program. We recommend that you either choose an account name and description like AUTOUSER or APIUSER, or one which is clearly connected with your company or application (e.g. MYCO or MYAPP). Note that you do not need to create a network account unless you require it for enhanced security - the Zetafax account does not need to be linked to a network login, though doing so will allow the Zetafax client to login automatically when started.

The installation steps required for the developer tools depend on the method you intend to use to integrate your application with the API, as follows:

COM API and C language libraries:
The Zetafax API is installed as part of the Zetafax API install to \zfax\ZFAPI. The installed folders contain libraries and include files; you should add the appropriate folders to your development tools (IDE).

API library functions communicate directly with the Zetafax server, using shared files. The API determines the location of the server through the presence of two files - either a file called ZETAFAX.INI, typically stored in the Program Files\Zetafax Server folder, or a file called ZFCLIENT.INI, typically stored either in the Program Files\Zetafax folder or the user's Application Data folder. ZETAFAX.INI is created when the Zetafax Server is installed. ZFCLIENT.INI is generated when the Zetafax client is run for the first time; however the file can sometimes be copied from another computer if preferred. In addition, registry values can be added to help the API locate these files. These registry values are:

Value Name	Registry Location	Description
ZetafaxIniPath	HKEY_LOCAL_MACHINE\Software\Equisys\Zetafax Server	String value pointing to ZETAFAX.INI file path (e.g. c:\zetafax\server)
ZetafaxClientIniPath	HKEY_LOCAL_MACHINE\Software\Equisys\Zetafax	String value pointing to ZFCLIENT.INI file path (e. g. c:\zetafax\client)

ZSUBMIT:
ZSUBMIT is an automatic submission program which is run automatically as part of the Zetafax server. It scans a single folder for messages to be sent - this is *zfax* \SERVER\Z-TEMP by default, though the location can be changed in the Zetafax Configuration program.

To use ZSUBMIT from your application there are no specific installation requirements, other than to ensure

that the server is configured to look in the folder your application will use.

Embedded addressing:
Embedded addressing commands are handled by the Zetafax Printer and Zetafax client program. The Zetafax client must be installed on the application computer, and the Zetafax server and Zetafax client must be running before printing to the Zetafax Printer (though the Zetafax Printer will start the client automatically if set to login automatically).

DDE:
DDE commands are handled by the Zetafax client program. The Zetafax client must be installed on the application computer, and the Zetafax server and Zetafax client must be running before initiating a DDE conversation.

3) Register the Zetafax API object model library (if you have not installed the Zetafax client)

If you intend to use the COM API (e.g. from a Visual Basic program), then you will need to register the object model library on the development computer. You do this from a command prompt as follows: Change directory to the location of ZFAPI32.DLLType the command REGSVR32 ZFAPI32.DLL A dialog box should appear confirming that the registration was successful.
Note: If you have installed the Zetafax client onto your development computer, the COM API is installed and registered automatically. It is installed by default to C:\Program Files\Common Files\Equisys.

4) Develop and test your custom application

Develop your application as normal. We recommend that your test cases include a variety of error scenarios such as number busy retries; these will depend on the degree and method of integration used, and the extent to which the application is intended to cope with normal errors unattended.

For test purposes, devices on the Zetafax server can be configured in demonstration mode. In this mode of operation the server simulates sending messages without connecting to the device. This removes the need for external connections and destination devices, and reduces the cost for high volume testing.

You configure a device for demonstration mode by editing the configuration file SETUP.INI (located in *zfax* \SYSTEM\Z-DB). Find the section for the device towards the end of the file (e.g. [FCLASS-1]), then add the following line at the end of the section:

DemoMode: SLOW SEND

Now restart the Zetafax server, and check that the device controller reports Demonstration mode enabled as it starts.

Deploying your application at a customer site

1) Obtain a Zetafax API product copy

When you are ready to deploy your fax-enabled solution to your customer, you will need to purchase the appropriate API product for your application from Equisys. You should also ensure that your customer has the appropriate hardware i.e. a PC server, fax hardware and telephone lines available prior to installation.

2) Install the Zetafax API

The Zetafax API typically includes a 1 user, 1 line Zetafax server, with special licensing to enable the API options. The API products include a CD-Rom with the software and a license number. Please refer to the documentation supplied in your developer kit or on the API CD-ROM for installation instructions. During installation, the Zetafax server software will be automatically registered with Equisys.

Create a Zetafax user account to use for sending messages, using the Zetafax Configuration program. This will typically have the same account name used when developing the application (see above), since applications written using the COM API, C language libraries or ZSUBMIT may specify the user name when they run.

Test that the API is correctly configured by sending and receiving a fax using the client software. Instructions are supplied in the Software Guide supplied with your Developer Kit or on the CD-ROM.

3) Install your custom application, including redistributable Zetafax API components

After installing your application you may need to carry out additional steps, depending on the method your application uses to integrate with the API, as follows:

COM API and C language libraries:
Install the Zetafax client program (*zfax* \SYSTEM\WKSETUP.EXE), this will create a ZFCLIENT.INI file and also install the ZFAPI32.DLL.

Or

Alternatively, you can create the ZFCLIENT.INI file by copying it from another computer if preferred and copy the following redistributable file from the API Developer kit:

File ZFAPI32.DLL, to be copied to %windir%\SYSTEM32

ZSUBMIT:
There are no specific installation requirements, other than to ensure that the server is configured to look in the folder your application will use.

Embedded addressing:
If your custom application is not running on the Zetafax server computer, install the Zetafax client by running the Zetafax Workstation Setup program (*zfax* \SYSTEM\WKSETUP.EXE).

DDE:
If your custom application is not running on the Zetafax server computer, install the Zetafax client by running the Zetafax Workstation Setup program (*zfax* \SYSTEM\WKSETUP.EXE). You may also wish to set the Zetafax client to startup automatically, by copying it to the Startup program group.

4) Register the ZFAPI.DLL (if you are using COM API and have not installed the Zetafax client)

If your custom application uses the COM API (e.g. from a Visual Basic program), then you will need to register the object model library on the server on which your custom application is installed. You can do this manually from a command prompt as follows:
Change directory to the location of ZFAPI32.DLL (i.e. %windir%\SYSTEM32)Type the command REGSVR32 ZFAPI32.DLL A dialog box should appear confirming that the registration was successful.

Alternatively you can do this automatically by calling REGSVR32 from your install program.

Note: If you have installed the Zetafax client, the COM API is installed and registered automatically. It is installed by default to C:\Program Files\Common Files\Equisys.

Contact and Support

If you require assistance with using or operating the software, please follow this procedure:

1. Read the on-line help manuals.
2. Search the support pages, and especially the technical notes section, on the Equisys web site at: http://www.equisys.com/support
3. Contact the software supplier from whom you purchased the software. In most cases the supplier will be able to provide support.

Please note that support is only available in selected countries, and you should contact your distributor for availability.

Please feel free to contact Equisys (or any of our distributors) with any comments or suggestions you may have about the software or the manual.

Equisys plc

http://www.equisys.com

Sales

Tel (020) 7203 4001
Fax (020) 7203 4005
sales@equisys.com

Technical support

Tel (020) 7203 4002
Fax (020) 7203 4005
support@equisys.com

Equisys Inc (USA and Canada)

Sales

Tel (770) 772 7201
Fax (770) 442 5789
sales@zetafax.com

Technical support

Tel(678) 942 7250
Fax(770) 442 5789
support@zetafax.com

Frequently Asked Questions

Q: What is the difference between the COM API and the C API?
A: The COM API is a wrapper for the C API. It adds a hierarchical structure to the function calls, and represents the API's complex structures as collections of objects. It was written to make the API more accessible from people writing applications with scripting languages and Visual Basic.

Q: Can you show me some code?
A: Using the COM API from within Visual Basic to send a high priority fax in 9 lines:

```
' Declare objects

Dim oZfAPI As New ZfLib.ZfAPI
Dim oUserSession As ZfLib.UserSession
Dim oNewMessage As ZfLib.NewMessage

' Logon and create new message:

Set oUserSession = oZfAPI.Logon("JJONES", False)
Set oNewMessage = oUserSession.CreateNewMsg

' Set properties:

oNewMessage.Recipients.AddFaxRecipient "Sam Smith", _"ACME plc", "020 7123 4567"
oNewMessage.Text = "I am a fax!"
oNewMessage.Priority = zfPriorityUrgent

' Send!

oNewMessage.Send
```

Q: What can you do with the COM API?
A: You can use the API to send faxes and to enumerate sent and received faxes. It can also be used to stop and start and gather status information about the Zetafax server, its devices and its LCR links.

Q: What can you not do with the COM API?
A: You cannot use the COM API to add, edit or delete users, or change the Zetafax server's configuration.

Q: With which development environments can the COM API be used?
A: It is compatible with all development environments which support COM, e.g. Visual Basic 6, Visual C++ 5 and 6, and Visual Studio .NET.

Q: Can you use the COM API with ASP?
A: Yes, but if the Zetafax server is installed on another computer you need to make sure that the web site is running as a user with permissions to access network drives, not the IWAM, IUSR or ASPNET accounts. (On Windows 2000 this could be done with COM+).

Q: In what language is the COM API written?
A: The API is written in C, the COM API is written in C++.

Q: With which operating systems is the COM API compliant?
A: The COM API is supported on all operating systems supported by the Zetafax client. (95, 98, Me, NT4, W2K, W2K3 and XP).

Q: Does the COM API work with managed code?
A: Yes. Once the COM API is registered you need to import it as a reference.

Q: Is the API thread-safe?
A: No. If you are using the API in a multi-threaded application it is best to protect all access to the Zetafax API with a critical section.

Q: How do you secure the COM API?
A: Zetafax stores user's files on the file system. To secure Zetafax you use NT security to only allow certain people access to each user's files.

Q: How does the COM API handle authentication?
A: You have to log on as a Zetafax user to user the COM API. To stop someone logging on to another user's account you have to use NT security on your file system.

Q: Which file formats are supported by the COM API?
A: By default, the API only supports text, G3N, G3F and some standard graphics formats. (Such as GIFs or Bitmaps). However, if you have the document rendering add-on, Doctiff, you can use over 200 other file types as well (see ZTN1261 - INFO File types supported by document conversion add on DOCTIFF).

Q: Can the COM API handle files as streams?
A: No, the API assumes that all files are already stored on the file system.

Q: Does the COM API support events?
A: No, you have to poll for changes.

Q: Can I use the COM API to send SMS messages?
A: Yes, however in the current version of the COM API the method is marked as hidden in the type library.

See ZTN1239 - HOWTO Develop and redistribute your fax-enabled application for details.

References
ZTN1239 - HOWTO Develop and redistribute your fax-enabled application
ZTN1261 - INFO File types supported by document conversion add on DOCTIFF

COM API

This help provides assistance on the COM API and the Zetafax Object Model.

For a brief introduction please read the Overview.

The details of the individual objects can be found in the ZfLib library.

Overview

Introduction

This document tells you how to configure and use the COM API.

Registration

Like all COM components ZfAPI32.dll must be registered before it can be used. This is done using the program regsvr32 as follows:

1. Change directory to location of ZfAPI32.dll
2. Run %system32%\regsvr32 ZfAPI32.dll
3. A dialog box should appear confirming that the registration was successful.

Using the COM API in Visual Basic

The COM API is primarily intended for people using Visual Basic. There are two ways of doing this:

- The recommended way is to add the Zfapi32 type library as a reference to your project. This enables early binding and access to all the constants and interfaces:

```
' Create new Zetafax object and Logon
Dim oZfAPI as New ZfLib.ZfAPI
Dim oUserSession as ZfLib.UserSession
Set oUserSession = oZfAPI.Logon("ADMINIST", False)
```

- You can also use the IDispatch interface and create objects using CreateObject and the ProgID:

```
' Create new Zetafax OBJECT USING CreateObject and Logon
DIM oZfAPI AS OBJECT
DIM oUserSession AS OBJECT
Set oZfAPI = CreateObject("ZfAPI32.ZfAPI")
Set oUserSession = oZfAPI.Logon("ADMINIST", FALSE)
```

Using the COM API in Visual C++

There are three ways of accessing COM objects in Visual C++:

- Win32 API. (Only for those who like to do it the hard way!)
- MFC OLE can be used to generate class wrappers for the ZfAPI32 type library using the Class Wizard.
- People using version 5.0 and later of Visual C++ can use the #import compiler directive to import ZfAPI32.dll. The compiler generates header files containing classes wrapping the interfaces, smart pointers, and the typdefs for the enumerations. This is the recommended way to use the COM API in Visual C++.

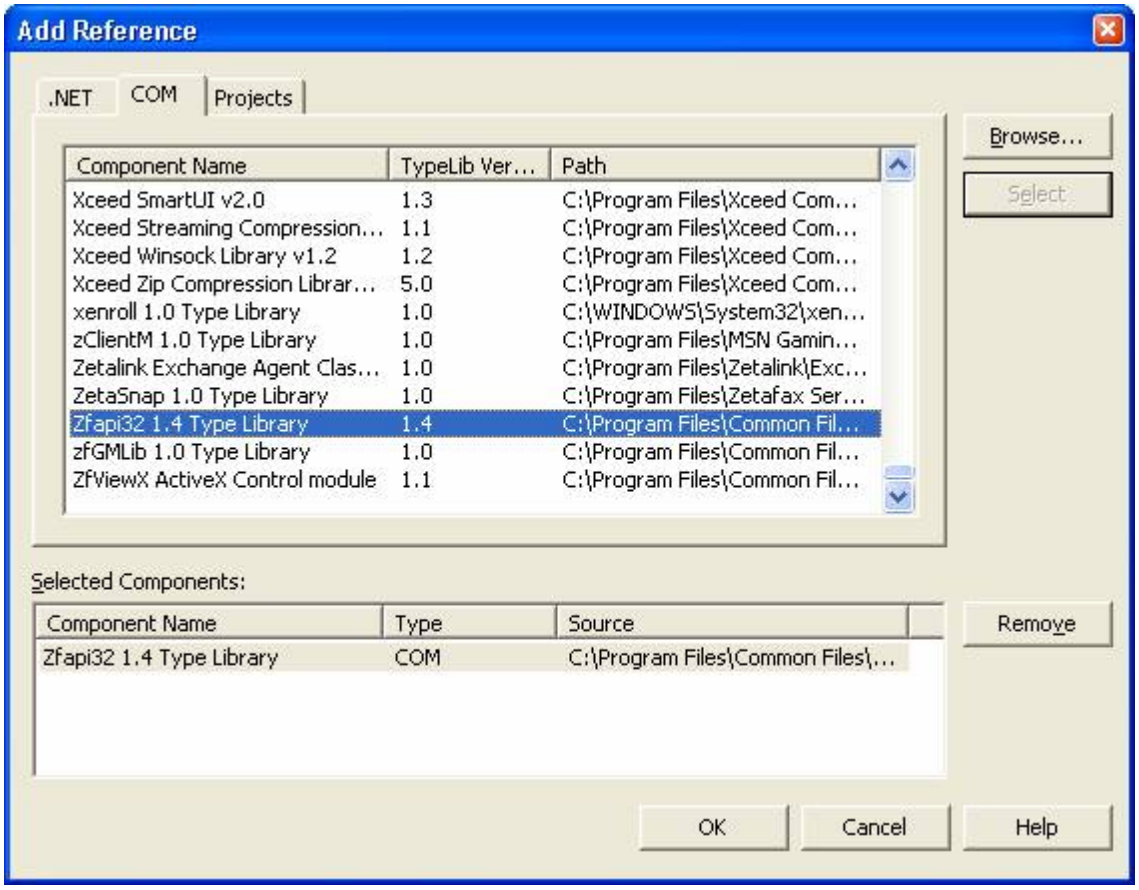
For example:

```
// Import Zetafax DLL without the ZfLib  namespace this will
// generate two header files, ZfAPI32.tlh and ZfAPI32.tli,
// which contain the definitions of the interfaces and enums
#import "ZfAPI32.dll" no_namespace

void DoSomething()
{
       // Create new Zetafax object and Logon
       IZfAPIPtr pZfAPI(_T("ZfAPI32.ZfAPI"));
```

```
        IZfUserSessionPtr pUser(pZfAPI->Logon(_T("ADMINIST"), false);
        ...
}
```

Using the COM API in .NET

1. From within Visual Studio select the add a reference option. The following menu will appear:



2. Select the COM tab option

3. Move to the end of the list, and search for the Zfapi32 1.4 Type library component. Select it by ensuring the component is selected and clicking on the Ok button.

Should the Zfapi32 1.4 Type library component not be present, then select the browse option to search for the *.dll file. In a standard Zetafaxsetup, this file is found in the following location:

C:\Program Files\Zetafax Server\ZFAPI\LIB\I386\Microsoft

4. You can now call functions present in the Zetafax library, by using references to the Zflib namespace within your code for example:

ZfLib.ZAPI

5. Ensure you distribute the interop.zflib.dll file found in the build directory of your code.

Code Samples

The following samples are written in C# and Visual Basic .NET. The other .NET languages (Managed C++, Visual J#) are very similar and these examples can easily be adapted for use in these environments.

Note: To avoid the usage of long type-names declare the following namespace at the top of the file:
using ZfLib;

The API fully supports exception handling and it is recommended that you use the try-catch mechanism to receive meaningful error messages and descriptions from the Zetafax COM API . (The examples below demonstrate this).

1. The following example demonstrates sending a fax using the COM API:

VB.NET:

```
' Declare objects
     Dim oZfAPI As New ZfLib.ZfAPI
     Dim oUserSession As ZfLib.UserSession
     Dim oNewMessage As ZfLib.NewMessage

     Try

         ' Logon and create NewMessage:
         oUserSession = oZfAPI.Logon("ADMINIST", False)
         oNewMessage = oUserSession.CreateNewMsg

         ' Set properties:
         oNewMessage.Recipients.AddFaxRecipient("Sam Smith", _
                                                "ACME plc", _
                                                "020 7123 4567")
         oNewMessage.Text = "I am a fax!"
         oNewMessage.Priority = ZfLib.PriorityEnum.zfPriorityUrgent

         ' Send!
         oNewMessage.Send()

     Catch ex As Exception
         System.Windows.Forms.MessageBox.Show(ex.Message,
    "ZetaFax Error",
    MessageBoxButtons.OK,
    MessageBoxIcon.Error)
     End Try
```

C#:

```
// Declare objects
ZfAPIClass oZfAPI = new ZfAPIClass();
UserSession          oUserSession;
ZfLib.NewMessage oNewMessage;

try
{
       // Logon and create NewMessage:
       oUserSession = oZfAPI.Logon("ADMINIST", false);
       oNewMessage  = oUserSession.CreateNewMsg();

       // Set properties:
       oNewMessage.Recipients.AddFaxRecipient("Sam Smith",    //TO
                                              "ACME plc",       //
Organization
                                              "020 7123 4567");//Fax number
       oNewMessage.Text = "I am a fax!";
       oNewMessage.Priority = ZfLib.PriorityEnum.zfPriorityUrgent;
```

```
            // Send!
            oNewMessage.Send();
        }
        catch (System.Runtime.InteropServices.COMException ex)
        {
            System.Windows.Forms.MessageBox.Show(ex.Message,
"ZetaFax Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
```

2. The code below demonstrates message information handling:

VB.NET:

```
        ' Declare objects:
      Dim strBody As String
      Dim oZfAPI As New ZfLib.ZfAPI
      Dim oUserSession As ZfLib.UserSession
      Dim oMessage As ZfLib.Message
      Dim oMsgHist As ZfLib.MessageHistory

      Try
          strBody = "~ZAPI001"

          ' Logon and get message:
          oUserSession = oZfAPI.Logon("ADMINIST", False)
          oMessage = oUserSession.Outbox.GetMsg(strBody)

          ' Display information about the message
          With oMessage.GetMsgInfo()
              txtCaption.Text = .Body
              txtSubject.Text = .Subject
              txtComment.Text = .Comment
          End With

          ' Display the number and call duration for each recipient:
          Dim MessageHistoryEnum As IEnumerator
          MessageHistoryEnum = oMessage.GetMsgHistories.GetEnumerator()

          While MessageHistoryEnum.MoveNext
              oMsgHist = MessageHistoryEnum.Current
              lstHistory.Items.Add("To: " & oMsgHist.Name & _
                       " Time Taken:" & oMsgHist.Date.ToString())
          End While

      Catch ex As Exception
          System.Windows.Forms.MessageBox.Show(ex.Message,
"ZetaFax Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
      End Try
```

C#:

```
            // Declare objects:
            string strBody;
            ZfAPIClass oZfAPI = new ZfAPIClass();
            UserSession oUserSession;
            ZfLib.Message oMessage;
            ZfLib.MessageHistory oMsgHist;
```

```
            try
            {
                  strBody = "~ZAPI001";

                  // Logon and get message:
                  oUserSession = oZfAPI.Logon("ADMINIST", false);
                  oMessage = oUserSession.Outbox.GetMsg(strBody);

                  // Display information about the message
                  txtCaption.Text = oMessage.GetMsgInfo().Body;
                  txtSubject.Text = oMessage.GetMsgInfo().Subject;
                  txtComment.Text = oMessage.GetMsgInfo().Comment;

                  // Display the number and call duration for each recipient:
                  string szFormattedHistoryItem;
            IEnumerator MessageHistoryEnum = oMessage.GetMsgHistories().GetEnumerator
      ( );

                  while (MessageHistoryEnum.MoveNext())
                  {
                        oMsgHist = (ZfLib.MessageHistory) MessageHistoryEnum.
      Current;
                        szFormattedHistoryItem = string.Format("To:{0} TimeTaken:
      {1}",
                        oMsgHist.Name,
                        oMsgHist.Date.ToString());
                        lstHistory.Items.Add(szFormattedHistoryItem);
                  }
            }
            catch (System.Runtime.InteropServices.COMException ex)
            {
                  System.Windows.Forms.MessageBox.Show(ex.Message,
      "ZetaFax Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
```

3. The following code demonstrates how to retrieve Zetafax device information:

VB.NET:

```
            ' Declare objects:
           Dim oZfAPI As New ZfLib.ZfAPI
           Dim oUserSession As ZfLib.UserSession
           Dim oDevices As ZfLib.Devices
           Dim oDevice As ZfLib.Device

           Try
               ' Logon and get devices:
               oUserSession = oZfAPI.Logon("ADMINIST", False)
               oDevices = oUserSession.Server.GetServerInfo().Devices

               ' Enumerate devices adding information to Listbox:
               Dim DeviceEnum As IEnumerator
               DeviceEnum = oDevices.GetEnumerator()

               While DeviceEnum.MoveNext
```

```
                'Iterate through the devices
                oDevice = DeviceEnum.Current
                lstDevices.Items.Add(oDevice.Name & " " & oDevice.User)
            End While
        Catch ex As Exception
            System.Windows.Forms.MessageBox.Show(ex.Message,
"Zetafax Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
        End Try
```

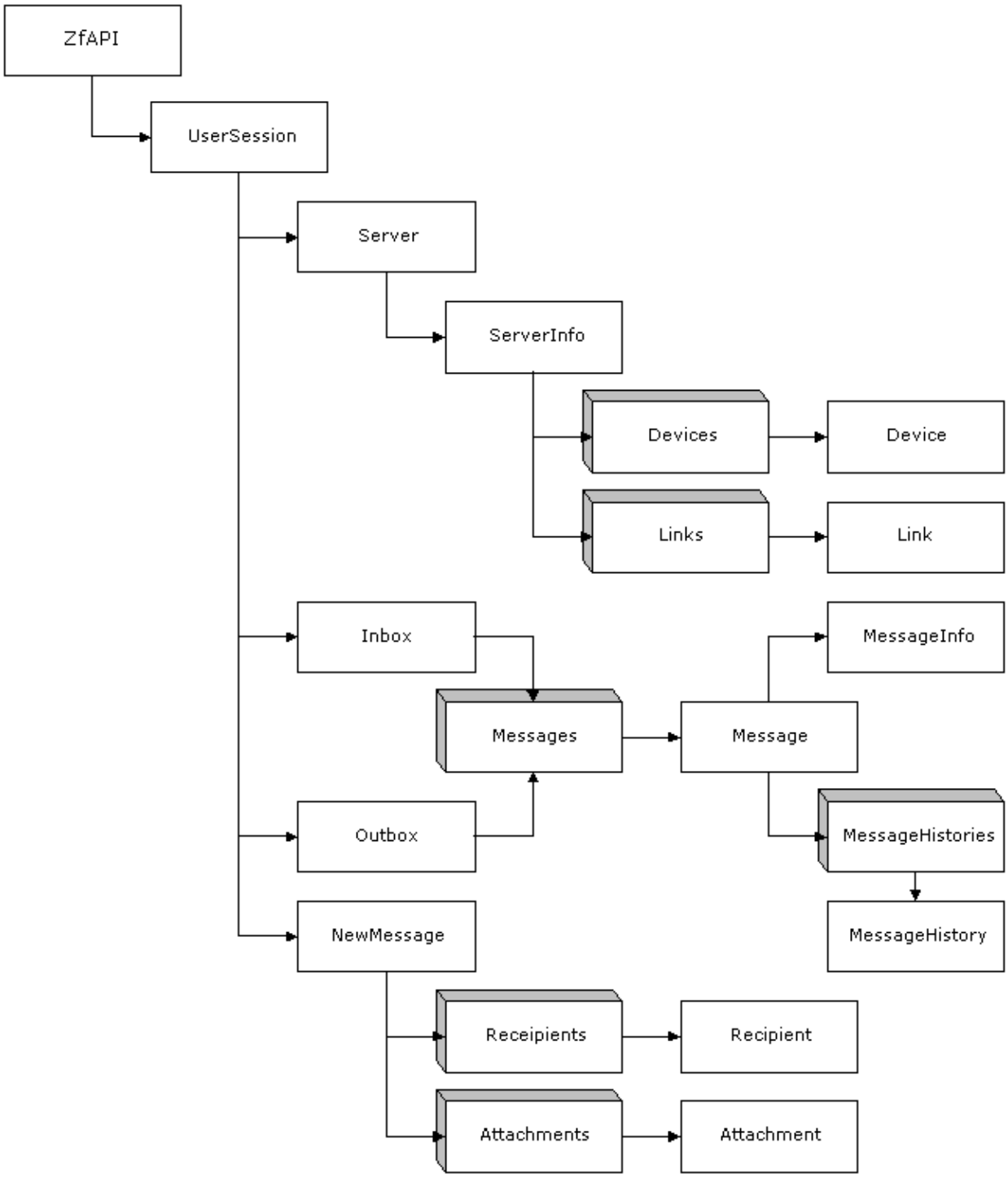
C#:

```
// Declare objects:
ZfAPIClass oZfAPI = new ZfAPIClass();
UserSession         oUserSession;
ZfLib.Devices oDevices;
ZfLib.Device  oDevice;

try
{
        // Logon and get devices:
        oUserSession = oZfAPI.Logon("ADMINIST", false);
        oDevices = oUserSession.Server.GetServerInfo().Devices;

        // Enumerate devices adding information to Listbox:
        IEnumerator DeviceEnum = oDevices.GetEnumerator();

        while(DeviceEnum.MoveNext())
        {
                oDevice = (ZfLib.Device)DeviceEnum.Current;
                lstDevices.Items.Add(oDevice.Name + " " + oDevice.User);
        }
}
catch (System.Runtime.InteropServices.COMException ex)
{
        System.Windows.Forms.MessageBox.Show(ex.Message,
        "ZetaFax Error",
        MessageBoxButtons.OK,
        MessageBoxIcon.Error);
}
```

The Zetafax Object Model

This diagram shows the various objects in the Zetafax object model, and how they are linked. Normal rectangles represent objects, shaded rectangles represent collections.

The ZfAPI object is the only object that can be created directly. All others can only be accessed by traversing the object tree.



Examples of the use of the COM API

Add the name of each device and the person using it into a list box:

```
' Declare objects:

Dim oZfAPI As New ZfLib.ZfAPI
Dim oUserSession As ZfLib.UserSession
Dim oDevices As ZfLib.Devices
```

```
Dim oDevice As ZfLib.Device

' Logon and get devices:
Set oUserSession = oZfAPI.Logon("ADMINIST", False)
Set oDevices = oUserSession.Server.ServerInfo.Devices

' Enumerate devices adding information to Listbox:
For Each oDevice In oDevices
       DevicesList.AddItem oDevice.Name & " " & oDevice.User
Next
```

Display information about a specific message:

```
' Declare objects:
Dim strBody As String
Dim oZfAPI As New ZfLib.ZfAPI
Dim oUserSession As ZfLib.UserSession
Dim oMessage As ZfLib .Message
Dim oMsgHist As ZfLib.MessageHistory

strBody = "~ZAPI001"

' Logon and get message:
Set oUserSession = oZfAPI.Logon("ADMINIST", False)
Set oMessage = oUserSession.Outbox.GetMsg(strBody)

' Display information about the message
With oMessage
       msgForm.Caption = .Body
       txtSubject = .Subject
       txtComment = .Comment
End With

' Display the number and call duration for each recipient:
For Each oMsgHist
In oMessage.GetMsgHistories
       lstHistory.AddItem
       "To: " & oMsgHist.AddrNum & _
              " Time Taken:"      & oMsgHist.Connection
Next
```

Send a fax:

```
' Declare objects
Dim oZfAPI As New ZfLib.ZfAPI
Dim oUserSession As ZfLib.UserSession
Dim oNewMessage As ZfLib.NewMessage

' Logon and create NewMessage:
Set oUserSession = oZfAPI.Logon("ADMINIST", False)
Set oNewMessage = oZfAPI.CreateNewMsg

'Set properties:
oNewMessage.Recipients.AddFaxRecipient "Sam Smith", _
       "ACME plc", "020 7123 4567"
oNewMessage.Text = "I am a fax!"
oNewMessage.Priority = zfPriorityUrgent

' Send!
oNewMessage.Send
```

Error Codes

This document explains what various errors returned by the COM API mean:

Error Name	Zetafax Error	Dispatch Error	COM error	Error description
E_INVALID_PARAM	0xF700	62720	0x8004F700	An invalid parameter was passed to the Zetafax
APIE_NOT_INIT	0xF701	62721	0x8004F701	A call was made to the API when it had not been initialised with a call to ZfAPI. Logon
E_INIT_FAIL	0xF702	62722	0x8004F702	The API failed to initialise
E_INVALID_ZF_INIT_FILE	0xF703	62723	0x8004F703	The API had problems reading ZETAFAX.INI
E_INVALID_ZF_USER	0xF704	62724	0x8004F704	An attempt was made to log on to the API with an invalid user
E_CANNOT_LOG_ON	0xF705	62725	0x8004F705	The user was already logged on or the API cannot access their directories
E_PATH_NOT_FOUND	0xF706	62726	0x8004F706	The API was passed an invalid path
E_TOO_MANY_FILES	0xF707	62727	0x8004F707	There are too many files in this directory
E_FILE_CREATE_ERROR	0xF708	62728	0x8004F708	Too many open files
E_FILE_OPEN_ERROR	0xF709	62729	0x8004F709	Could not open file
E_FILE_ERROR	0xF70A	62730	0x8004F70A	Could not read/write to file
E_FILE_NOT_FOUND	0xF70B	62731	0x8004F70B	The file could not be found
E_SERVER_NOT_RUNNING	0xF70C	62732	0x8004F70C	The Zetafax server isn't running
E_INVALID_DATA_FORMAT	0xF70D	62733	0x8004F70D	The data file format specified is not recognised
E_MSG_UNKNOWN	0xF70E	62734	0x8004F70E	The specified message could not be found
E_MSG_NOT_COMPLETED	0xF70F	62735	0x8004F70F	An attempt was made to modify a message that wasn't completed
E_MSG_EXISTS	0xF710	62736	0x8004F710	This message already exists
E_INFO_FILE_OPEN_ERROR	0xF711	62737	0x8004F711	Could not open MSGDIR.CTL
E_INFO_FILE_ERROR	0xF712	62738	0x8004F712	Error updating/accessing MSGDIR.CTL
E_INFO_FILE_INVALID	0xF713	62739	0x8004F713	MSGDIR.CTL is invalid
E_CANNOT_SUBMIT	0xF714	62740	0x8004F714	Error handling .SUB file
E_BUFFER_TOO_SMALL	0xF715	62741	0x8004F715	The buffer in which to return data was too small
E_SUBMIT_FILE_INVALID	0xF716	62742	0x8004F716	One or more lines in the specified .SUB file were invalid
E_CANNOT_READ_MSG_DEFAULTS	0xF717	62743	0x8004F717	Cannot read user's USER.INI
E_CONTROL_FILE_OPEN_ERROR	0xF718	62744	0x8004F718	The API was unable to open the message's CTL file
E_CONTROL_FILE_	0xF719	62745	0x8004F719	Error reading/writing CTL file

ERROR				
E_CONTROL_FILE_INVALID	0xF71A	62746	0x8004F71A	CTL file is corrupt
E_SERVER_RUNNING	0xF71B	62747	0x8004F71B	The Zetafax server is running
E_NO_START_SYSMAN	0xF71C	62748	0x8004F71C	The API could not launch SYSMAN.EXE
E_SERVER_INI_FILE_ERROR	0xF71E	62750	0x8004F71E	Problems accessing SETUP.INI
E_FUNCTION_ABORT	0xF71F	62751	0x8004F71F	The function failed because a user defined callback function returned "Abort and Stop" status
E_TIMEOUT_EXPIRED	0xF720	62752	0x8004F720	The specified function did not complete within the timeout period
E_MALLOC_FAILED	0xF724	62756	0x8004F724	The API ran out of memory.
E_LICENCE_ERROR	0xF725	62757	0x8004F725	The API Could not find the Zetafax licence
E_API_LICENSE_ERROR	0xF726	62758	0x8004F726	Your Zetafax licence does not permit you to use the API
E_USER_NOT_INIT	0xF727	62759	0x8004F727	The API couldn't find the user of this session
E_ACCESSDENIED	0xF728	62760	0x8004F728	Access Denied
E_FULLCOLLECTION	0xF730	62768	0x8004F730	The collection is full

ZfLib

This is a list of the objects and enumerations that appear in the Zetafax COM library

Groups

COM Classes

API Print	This object allows the API to print via the Zetafax Printer.
Attachment	The Attachment object contains the name of a Zetafax attachment to be attached to a message before it is sent.
Attachments	The Attachments collection object contains Attachment objects.
Coversheet	The Coversheet class represents a Zetafax coversheet.
Coversheets	The Coversheets collection object contains Coversheet objects.
Device	The Device object contains the configuration information and status of a device attached to the Zetafax server.
Devices	The Devices collection contains Device objects.
File	The File object contains the path of a file to be attached to the message before it is sent.
Files	The Files collection object contains File objects. To attach Zetafax attachments to the message use the Attachments collection.
Inbox	The Inbox object represents a user's Zetafax IN directory.
Letterhead	The Letterhead class represents a Zetafax letterhead.
Letterheads	The Letterheads collection object contains Letterhead objects.
Link	The Link object allows a user to retrieve information on the status of a link. It also provides statistics on the Link 's performance.
Links	The Links collection contains Link objects.
Message	The Message object allows access to the details of sent

	and received messages.
MessageHistories	The MsgHistories collection represents a message's transmission history and contains MessageHistory objects.
MessageHistory	The MessageHistory object contains an item in a message's transmission history.
MessageInfo	The MessageInfo object contains information about a single Message object
Messages	The Messages collection contains Message objects from either a User's Inbox or Outbox .
NewMessage	The NewMessage object allows the logged on user to prepare and submit a new message to the Zetafax server for sending.
Outbox	The Outbox object represents a user's Zetafax OUT directory.
Recipient	The Recipient object contains address details of an addressee for whom a message is intended.
Recipients	The Recipients collection object contains Recipient objects.
Server	The Server object allows control over the Zetafax server, and access to objects describing the server's state.
ServerInfo	The ServerInfo object exposes the status and configuration of the Zetafax server.
UserSession	The UserSession object contains details about a Zetafax user's configuration, messages and gives you the ability to create new messages for sending.
ZfAPI	The ZfAPI object is the Application object of the COM version of the API, and is the only object that can be created directly. You must create this object and logon before you can access any of the other objects.

Type Definitions

DevStatusEnum	The DevStatusEnum enumeration specifies the current status of a Device.
EventEnum	The EventEnum enumeration specifies the event described in

	a MessageHistory object
FaxTypeEnum	The FaxTypeEnum enumeration specifies how a recipient will be sent a message
FormatEnum	The FormatEnum enumeration specifies the format of the data file to be sent
HeaderEnum	This HeaderEnum Enumeration specifies the content of the header line to appear in the top page of each fax
LinkStatusEnum	The LinkStatusEnum enumeration specifies the current status of an LCR link
PriorityEnum	The PriorityEnum enumeration specifies the priority of an outgoing message
QualityEnum	The QualityEnum enumeration specifies the resolution when sending a fax
RouteEnum	The RouteEnum enumeration specifies the type of route used
SendTimeEnum	The SendTimeEnum enumeration specifies when a message will be sent
StatusEnum	The StatusEnum specifies the current status of a message
UserStatusEnum	The StatusEnum specifies the 'user status' of a message (that is the user's awareness of the message)
ZfErr	The ZfErr enumeration specifies the errors returned the API.

ZfLib.APIPrint

This object allows the API to print via the Zetafax Printer.

Groups

Operations

CancelPrint	This method cancels the API print mode.
IsComplete	This method checks if the Zetafax Printer has completed spooling the print output file.
StartPrint	This method switches the Zetafax Printer into API Print mode.

Read-only Properties

FileName	Returns the print spool filename.

ZfLib.APIPrint.CancelPrint

```
Sub CancelPrint( )
```

This method cancels the API print mode.

Remarks

When the StartPrint method is called, the Zetafax Printer is set to API Print mode; the next job will be treated as an API print job. If you have called StartPrint but an error has occurred during printing, you should call CancelPrint to clear the API print mode. Any subsequent jobs sent to the Zetafax Printer will then be handled in the usual way (e.g. they will be sent to the Zetafax Client).

See Also
APIPrint.StartPrint ,
APIPrint.IsComplete

ZfLib.APIPrint.FileName

```
Property FileName As String
```

Returns the print spool filename.

Return Value

[out, retval]
The print spool filename

Remarks

This method returns the print spool filename that was set by StartPrint .
See Also

APIPrint.StartPrint

ZfLib.APIPrint.IsComplete

```
Function IsComplete( ) As Boolean
```

This method checks if the Zetafax Printer has completed spooling the print output file.

Return Value

Completion status

Remarks

This method works by checking the lock status on the output file. If the permissions on the file are not sufficient to open for writing, the method will return access denied.

ZfLib.APIPrint.StartPrint

```
Sub StartPrint( ByVal bszFileName As String )
```

This method switches the Zetafax Printer into API Print mode.

Parameters

bszFileName	Filename to print to

Remarks

Once this method has been called, the next print job sent to the Zetafax Printer under the current user context will be sent to the specified file. The Zetafax Client will not be launched. As soon as the Zetafax Printer begins processing the print job, the API print mode is cleared and subsequent jobs will be processed as normal.

See Also

APIPrint.IsComplete,
APIPrint.CancelPrint

ZfLib.Attachment

The Attachment object contains the name of a Zetafax attachment to be attached to a message before it is sent.

Groups

Operations

Delete	The Delete method removes an Attachment item from the Attachments collection.

Read/Write Properties

Name	The Name property returns the name of the attachment

ZfLib.Attachment.Delete

```
Sub Delete( )
```

The Delete method removes an Attachment item from the Attachments collection.

See Also
Attachments ,
NewMessage

ZfLib.Attachment.Name

```
Property Name As String
```

The Name property returns the name of the attachment

Return Value

[out, retval]

The name of the attachment

See Also
Attachments , NewMessage

ZfLib.Attachments

The Attachments collection object contains Attachment objects.

Remarks

This collection contains attachment objects. These are Zetafax graphics attachments. To add files to the message use the Files collection. The Attachments collection cannot contain more than 30 objects.

Groups

Operations

Add	The Add method adds an Attachment to the collection.

Read-only Properties

Count	The Count property returns the number of Attachment items in the collection.
Item	The Item property returns the specified Attachment object.

See Also

Attachment , Files

ZfLib.Attachments.Add

```
Function Add( ByVal bszFile As String ) As ZfLib.Attachment
```

The Add method adds an Attachment to the collection.

Parameters

bszFile

The name of the attachment to be added to the collection.

Return Value

The new Attachment object

Remarks

If the attachment is taken from the Z-GRAPH directory, and is identified by it's Zetafax name, not it's file name.

See Also

Attachment ,
NewMessage

ZfLib.Attachments.Count

```
Property Count As Long
```

The Count property returns the number of Attachment items in the collection.

Return Value

[out, retval]

The number of items in the collection

ZfLib.Attachments.Item

```
Property Item( ByVal var As Variant ) As ZfLib.Attachment
```

The Item property returns the specified Attachment object.

Parameters

var

[in]

The index of the Attachment object to retrieve from the collection.

Return Value

The Attachment object retrieved from the collection.

ZfLib.Coversheet

The Coversheet class represents a Zetafax coversheet.

Groups

Read-only Properties

Name — The Name property returns the name of the Coversheet.

ZfLib.Coversheet.Name

```
Property Name As String
```

The Name property returns the name of the Coversheet.

Return Value

[out, retval]

See Also
Coversheets

ZfLib.Coversheets

The Coversheets collection object contains Coversheet objects.

Groups
Read-only Properties

Count	The Count property returns the number of Coversheet items in the collection.
Item	The Item property returns a coversheet item from the collection.

See Also
Coversheet , ServerInfo

ZfLib.Coversheets.Count

```
Property Count As Long
```

The Count property returns the number of Coversheet items in the collection.

Return Value

[out, retval]

The number of coversheet items in the Coversheets collection

See Also

ServerInfo ,
Coversheet

ZfLib.Coversheets.Item

```
Property Item( ByVal var As Variant ) As ZfLib.Coversheet
```

The Item property returns a coversheet item from the collection.

Parameters


var

[in]

The index of the coversheet to retrieve from the collection.

Return Value

The Coversheet object retrieved from the Coversheets collection

Remarks

The Coversheets property needs to be called using the ServerInfo object to retrieve an updated version of the Coversheets collection.

Also See:
Coversheet ,
ServerInfo

ZfLib.Device

The Device object contains the configuration information and status of a device attached to the Zetafax server.

Groups

Read-only Properties

CurrentPage	The CurrentPage property returns the current page being sent by the device.
MsgBody	The MsgBody property returns the body of the file name of the control file currently

Name	being processed by the device. The Name property returns the name of the device. This comprises the Device Type and Device Number separated by a hyphen. An Example of a Device name might be FLASS-3.
NumConnectFails	The NumConnectFails property returns the number of send attempts by the device that failed to connect.
NumPages	The NumPages property returns the number of pages in the message currently being processed.
NumSendFails	The NumSendFails returns the number of send attempts that failed after the device connected successfully.
NumSendOK	The NumSendOK property returns the number of messages the device has sent successfully.
Status	The Status property returns the current status of the Device object.
User	The User property returns the Username of the owner of the message currently being processed by the device.

ZfLib.Device.CurrentPage

```
Property CurrentPage As Short
```

The CurrentPage property returns the current page being sent by the device.

Return Value

[out, retval]

The current page being sent by the device.

See Also
Devices

ZfLib.Device.MsgBody

```
Property MsgBody As String
```

The MsgBody property returns the body of the file name of the control file currently being processed by the device.

Return Value

[out, retval]

The body name of the message control file

Remarks

Message bodies are made up of a "~", a four letter identifier, and a three letter/digit unique number. An example of a message body might be ~ZAPI001.

Also See: Devices

ZfLib.Device.Name

```
Property Name As String
```

The Name property returns the name of the device. This comprises the Device Type and Device Number separated by a hyphen. An Example of a Device name might be FLASS-3.

Return Value

[out, retval]
The name of the device

Remarks

Also See: Devices

ZfLib.Device.NumConnectFails

```
Property NumConnectFails As Short
```

The NumConnectFails property returns the number of send attempts by the device that failed to connect.

Return Value

[out, retval]

The number of send attempts by the device that failed because of connection problems.

ZfLib.Device.NumPages

```
Property NumPages As Short
```

The NumPages property returns the number of pages in the message currently being processed.

Return Value

[out, retval]

The number of pages in the message currently being processed.

Remarks

The coversheet is included in the number of pages being sent.

ZfLib.MessageHistory.Route

```
Property Route As RouteEnum
```

The Route property returns the route type used to send a message.

Return Value

[out, retval]

This is the route type used to send a message.

ZfLib.Device.NumSendOK

Property

```
NumSendOK As Short
```

The NumSendOK property returns the number of messages the device has sent successfully.

Return Value

[out, retval]

The number of messages sent successfully on the device

ZfLib.Device.User

```
Property User As String
```

The User property returns the Username of the owner of the message currently being processed by the device.

Return Value

[out, retval]

The user configured to use the device

See Also
Devices

ZfLib.Devices

The Devices collection contains Device objects.

Remarks

The collection, and the items in the collection, represent a snap-shot of the Devices' status. If you want to update the collection you need to get a new ServerInfo object from the Server object.
Objects cannot be added or removed from this collection
This collection cannot exceed 100 objects.

Groups

Read-only Properties

Count	The Count property returns the number of Device items in the collection.
Item	The Item property returns a device item from the collection.

See Also
Device , ServerInfo , Server

ZfLib.Devices.Count

```
Property Count As Long
```

The Count property returns the number of Device items in the collection.

Return Value

[out, retval]

The number of device items in the Devices collection

See Also
ServerInfo , Device

ZfLib.Devices.Item

```
Property Item( ByVal var As Variant ) As ZfLib.Device
```

The Item property returns a device item from the collection.

Parameters

 var

[in]

The index of the device to retrieve from the collection.

Return Value

The Device object retrieved from the Devices collection

Remarks

The Devices property needs to be called using the ServerInfo object to retrieve the update version version of the Devices collection.

Also See: Device , ServerInfo

ZfLib.File

The File object contains the path of a file to be attached to the message before it is sent.

Groups

Operations

Delete	The Delete method removes the File object from the Files collection.

Read/Write Properties

FileName	The FileName property returns the path of the file to be attached.

ZfLib.File.Delete

```
Sub Delete( )
```

The Delete method removes the File object from the Files collection.

See Also
Files , NewMessage

ZfLib.File.FileName

```
Property FileName As String
```

The FileName property returns the path of the file to be attached.

Parameters

strFilename

[in]

Return Value

[out, retval]

The path of the file attached to a new message
See Also
Files , NewMessage

ZfLib.Files

The Files collection object contains File objects. To attach Zetafax attachments to the message use the Attachments collection.

Groups

Operations

Add	The Add method adds a File object to the collection.

Read-only Properties

Count	The Count property returns the number of File items in the collection.
Item	The Item property returns the specified File object.

See Also
File , Attachments

ZfLib.Files.Add

```
Function Add( ByVal bszFile As String ) As ZfLib.File
```

The Add method adds a File object to the collection.

Parameters

bszFile

The path of the file to be added to the collection.

Return Value

This method returns the new File object

Remarks

To reduce the possibility of errors, it is best to specify the full path when adding files to the collection.

See Also
File , NewMessage

ZfLib.Files.Count

```
Property Count As Long
```

The Count property returns the number of File items in the collection.

Return Value

[out, retval]

The number of items in the collection .

ZfLib.Files.Item

```
Property Item( ByVal var As Variant ) As ZfLib.File
```

The Item property returns the specified File object.

Parameters

var

[in]

The index of the File object to retrieve from the collection.

Return Value

The File object retrieved from the collection.

ZfLib.Inbox

The Inbox object represents a user's Zetafax IN directory.

Remarks

This object is the same as the Outbox object except that it points to a different folder.

Groups

Operations

CheckNewMsgStatus	The CheckNewMsgStatus method checks whether the status of any of the messages in the current user's IN directory have changed.
GetMsg	The GetMsg method returns the Message object matching the specified message body name.
GetMsgList	The GetMsgList method returns a Messages collection containing all

the items in this directory

See Also
Outbox , Messages, Message

ZfLib.Inbox.CheckNewMsgStatus

```
Function CheckNewMsgStatus( ) As Boolean
```

The CheckNewMsgStatus method checks whether the status of any of the messages in the current user's IN directory have changed.

Return Value

Returns True if the status of any of the messages have changed

See Also
ZfAPI , UserSession , Outbox

ZfLib.Inbox.GetMsg

```
Function GetMsg( ByVal bszMsg As String ) As ZfLib .Message
```

The GetMsg method returns the Message object matching the specified message body name.

Parameters

bszMsg

The body name of the message to retrieve.

See Also
Messages, Message

ZfLib.Inbox.GetMsgList

```
Function GetMsgList( ) As ZfLib .Messages
```

The GetMsgList method returns a Messages collection containing all the items in this directory

Return Value

The Messages collection object retrieved from the user's IN directory.

See Also
Outbox , Messages

ZfLib.Letterhead

The Letterhead class represents a Zetafax letterhead.

Groups

Read-only Properties

Name	The Name property returns the name of the Letterhead.

ZfLib.Letterhead.Name

```
Property Name As String
```

The Name property returns the name of the Letterhead.

Return Value

[out, retval]

See Also
Letterheads

ZfLib.Letterheads

The Letterheads collection object contains Letterhead objects.

Groups

Read-only Properties

Count	The Count property returns the number of Letterhead items in the collection.
Item	The Item property returns a letterhead item from the collection.

See Also
Letterhead , ServerInfo

ZfLib.Letterheads.Count

```
Property Count As Long
```

The Count property returns the number of Letterhead items in the collection.

Return Value

[out, retval]

The number of letterhead items in the Letterheads collection

See Also
ServerInfo , Letterhead

ZfLib.Letterheads.Item

```
Property Item( ByVal var As Variant ) As ZfLib.Letterhead
```
The Item property returns a letterhead item from the collection.

Parameters

var

[in]

The index of the letterhead to retrieve from the collection.

Return Value

The Letterhead object retrieved from the Letterheads collection

Remarks

The Letterheads property needs to be called using the ServerInfo object to retrieve the update version version of the Letterheads collection.

See Also
Letterhead , ServerInfo

ZfLib.Link

The Link object allows a user to retrieve information on the status of a link. It also provides statistics on the Link's performance.

Groups

Read-only Properties

ConnectionOK	The ConnectionOK property returns the current state of the connection to the Remote server.
LinkActive	The LinkActive property returns the active status of the Link.
LocalStatus	The LocalStatus property retrieves the current status of the link.
NumAcknowledged	The NumAcknowledged property returns the number of messages that have been submitted to the Remote server and have been acknowledged.
NumDeviceError	The NumDeviceError property returns the number of messages that failed to be sent as a result of device errors.
NumReceived	The NumReceived property returns the number of messages that have been received from the Remote server for sending locally.
NumRejected	The NumRejected property returns the number of messages rejected by the Remote server.
NumRemoteServerError	The NumRemoteServerError property returns the number of messages that failed to be sent as a result of other errors at the Remote server.
NumSentOK	The NumSentOK property returns the number of messages sent successfully by the Remote server.
NumTimedOut	The NumTimedOut property returns the number of messages that timed out while waiting for a response over the link.
NumUnAcknowledged	The NumUnAcknowledged property returns the number of messages that haven't been acknowledged.
RemoteServer	The RemoteServer property returns the name of the Remote server to which the link is configured.
RemoteStatus	The RemoteStatus property returns the current status of the remote server.

ZfLib.Link.ConnectionOK

```
Property ConnectionOK As Boolean
```

The ConnectionOK property returns the current state of the connection to the Remote server.

Return Value

[out, retval]

This property returns the state of the connection. True if it is OK, False otherwise.

See Also
Links

ZfLib.Link.LinkActive

```
Property LinkActive As Boolean
```

The LinkActive property returns the active status of the Link .

Return Value

[out, retval]

Returns True to indicate the link is active and available for sending and receiving, otherwise it returns False.

See Also
Links

ZfLib.Link.LocalStatus

Property

LocalStatus As LinkStatusEnum

The LocalStatus property retrieves the current status of the link.

Return Value

[out, retval]
This is the current status of the link on the local server.

See Also
Links , LinkStatusEnum

ZfLib.Link.NumAcknowledged

```
Property NumAcknowledged As Short
```

The NumAcknowledged property returns the number of messages that have been submitted to the Remote server and have been acknowledged.

Return Value

[out, retval]

The number of messages submitted that have been acknowledged.

See Also
Links

ZfLib.Link.NumDeviceError

```
Property NumDeviceError As Short
```

The NumDeviceError property returns the number of messages that failed to be sent as a result of device errors.

Return Value

[out, retval]

The number of device errors that occurred on the Remote server

See Also
Links

ZfLib.Link.NumReceived

```
Property NumReceived As Short
```

The NumReceived property returns the number of messages that have been received from the Remote server for sending locally.

Return Value

[out, retval]

The number of messages received from the Remote server

See Also
Links

ZfLib.Link.NumRejected

```
Property NumRejected As Short
```

The NumRejected property returns the number of messages rejected by the Remote server.

Return Value

[out, retval]

The number of messages rejected by the Remote server.

See Also
Links

ZfLib.Link.NumRemoteServerError

```
Property NumRemoteServerError As Short
```

The NumRemoteServerError property returns the number of messages that failed to be sent as a result of other errors at the Remote server.

Return Value

[out, retval]

The number of Remote server errors on this link

See Also

Links

ZfLib.Link.NumSentOK

```
Property NumSentOK As Short
```

The NumSentOK property returns the number of messages sent successfully by the Remote server.

Return Value

[out, retval]

The number of messages sent successfully by the Remote server

See Also

Links

ZfLib.Link.NumTimedOut

```
Property NumTimedOut As Short
```

The NumTimedOut property returns the number of messages that timed out while waiting for a response over the link.

Return Value

[out, retval]

The number of messages timed out on the link

ZfLib.Link.NumUnAcknowledged

```
Property NumUnAcknowledged As Short
```

The NumUnAcknowledged property returns the number of messages that haven't been acknowledged.

Return Value

[out, retval]

The number of messages awaiting acknowledgedment.

See Also
Links

ZfLib.Link.RemoteServer

Property

```
RemoteServer As String
```

The RemoteServer property returns the name of the Remote server to which the link is configured.

Return Value

[out, retval]
The name of the remote server

ZfLib.Link.RemoteStatus

```
Property RemoteStatus As LinkStatusEnum
```

The RemoteStatus property returns the current status of the remote server.

Return Value

[out, retval]

The current status of the Remote server

See Also
Links , LinkStatusEnum

ZfLib.Links

The Links collection contains Link objects.

Remarks

The collection, and the items in the collection, represent a snap-shot of the Links' status. If you want to update the collection you need to get a new ServerInfo object from the Server object.

Objects cannot be added to or removed from this collection.

Groups

Read-only Properties

Count	The Count property returns the total number of Link items in the collection.
Item	The Item property returns the specified Link object.

See Also
Link , ServerInfo , Server

ZfLib.Links.Count

```
Property Count As Long
```

The Count property returns the total number of Link items in the collection.

Return Value

[out, retval]

The number of Link items in the Links collection

See Also

Link , ServerInfo

ZfLib.Links.Item

```
Property Item( ByVal var As Variant ) As ZfLib.Link
```

The Item property returns the specified Link object.

Parameters

var

[in]

The index of the Link object to retrieve from the collection.

Return Value

The Link object retrieved from the Links collection.

See Also
Link , ServerInfo

ZfLib.Message

The Message object allows access to the details of sent and received messages.

Remarks

You can use this object to delay or rush messages that have not yet been sent.

To create a new message you have to use the NewMessage object returned by UserSession.CreateNewMsg .

Groups

Operations

AbortMsg	The AbortMsg method aborts the sending of a message.
DeleteMsg	The Delete method removes an entry of the message from the IN or OUT message information file.
GetMsgHistories	The GetMsgHistories method returns the MessageHistories collection containing the transmission history of the message.
GetMsgInfo	The GetMsgInfo method returns the MessageInfo object. This allows access to information about a message.
HoldMsg	The HoldMsg method holds the sending of the current message.
MarkMsgAsRead	The MarkMsgAsRead method changes the 'user status' of a message to 'OK'. This will make it appear as 'read' to client programs.
ReleaseMsg	The ReleaseMsg method releases a held message and places it in the message queue for sending.
RushMsg	The RushMsg method is used to rush a message for sending.
SendMsg	The SendMsg method places a message in the message queue for sending by the Zetafax server.

See Also
NewMessage

ZfLib.Message.AbortMsg

```
Sub AbortMsg( )
```

The AbortMsg method aborts the sending of a message.

See Also
Messages

ZfLib.Message.DeleteMsg

```
Sub DeleteMsg( ByVal fDeleteFiles As Boolean )
```

The Delete method removes an entry of the message from the IN or OUT message information file.

Parameters

fDeleteFiles

Boolean flag to delete associate data files if True.

Remarks

The Delete method removes the object from the collection. If you have another reference to the object it's properties and methods may return errors.

ZfLib.Message.GetMsgHistories

```
Function GetMsgHistories( ) As ZfLib.MessageHistories
```

The GetMsgHistories method returns the MessageHistories collection containing the transmission history of the message.

Return Value

The retrieved MessageHistories collection.

See Also
Messages

ZfLib.Message.GetMsgInfo

Function GetMsgInfo( ) As ZfLib.MessageInfo

The GetMsgInfo method returns the MessageInfo object. This allows access to information about a message.

Return Value

The retrieved MessageInfo object.

See Also
Messages

ZfLib.Message.HoldMsg

```
Sub HoldMsg( )
```

The HoldMsg method holds the sending of the current message.

See Also
Messages, Inbox

ZfLib.Message.MarkMsgAsRead

```
Sub MarkMsgAsRead( )
```

The MarkMsgAsRead method changes the 'user status' of a message to 'OK'. This will make it appear as 'read' to client programs.

See Also
Messages, Inbox

ZfLib.Message.ReleaseMsg

```
Sub ReleaseMsg( )
```

The ReleaseMsg method releases a held message and places it in the message queue for sending.

See Also
Messages, Inbox

ZfLib.Message.RushMsg

```
Sub RushMsg( )
```

The RushMsg method is used to rush a message for sending.

See Also
Messages

ZfLib.Message.SendMsg

```
Sub SendMsg( )
```

The SendMsg method places a message in the message queue for sending by the Zetafax server.

See Also
Messages, Inbox

ZfLib.MessageHistories

The MsgHistories collection represents a message's transmission history and contains MessageHistory objects.

Remarks

The collection, and the items in the collection, represent a snap-shot of the message's history. If you want to update the collection you need to get a new MessageHistories object from it's parent Message object.

Objects cannot be added or removed from this collection.

Groups

Read-only Properties

Count	The Count method returns the number of MessageHistory items in the collection.
Item	The Item property returns a MessageHistory item from the collection.

See Also
MessageHistory , Message, MessageInfo

equisys ZETAfax

ZfLib.MessageHistories.Count

```
Property Count As Long
```

The Count method returns the number of MessageHistory items in the collection.

Return Value

[out, retval]

The number of MessageHistory items in the collection.

See Also
Message, MessageHistory

equisys ZETAfax

ZfLib.MessageHistories.Item

```
Property Item( ByVal var As Variant ) As ZfLib.MessageHistory
```

The Item property returns a MessageHistory item from the collection.

Parameters

var

[in]

The MessageHistory item to retrieve from the MessageHistories collection.

Return Value

The MessageHistory object retrieved from the collection.

See Also
Message, MessageHistory

ZfLib.MessageHistory

The MessageHistory object contains an item in a message's transmission history.

Groups

Read-only Properties

AddrNum	The AddrNum property returns addressee number starting from 1. This allows events to be matched to addressees when a single message has been sent to more than one person.
Connection	The Connection property returns the connection time in seconds.
Date	The Date property returns the transmission date of the message.
Device	The Device property returns the name of the device used to send the message.
ErrorCode	The ErrorCode property returns the reason an attempt to send a message failed.
Event	The Event property returns the Event type of this MessageHistory object.
Name	The Name property returns the name of the recipient associated with this MessageHistory event
Organisation	The Organisation property returns the organisation of the recipient associated with this MessageHistory event
PagesSent	The PagesSent property returns the last page successfully sent.
RemoteServer	The RemoteServer property returns the name of the remote server used to send a message when the message was sent using LCR.
Route	The Route property returns the route type used to send a message.
RouteParams	The RouteParams property returns the route parameters used. For fax routes this is the fax number dialled.

ZfLib.MessageHistory.AddrNum

```
Property AddrNum As Short
```

The AddrNum property returns addressee number starting from 1. This allows events to be matched to

addressees when a single message has been sent to more than one person.

Return Value

[out, retval]

The addressee index.

See Also
Messages, MessageHistories

ZfLib.MessageHistory.Connection

```
Property Connection As Short
```

The Connection property returns the connection time in seconds.

Return Value

[out, retval]

The connection time in seconds

See Also

Messages, MessageHistories

ZfLib.MessageHistory.Date

```
Property Date As Date
```

The Date property returns the transmission date of the message.

Return Value

[out, retval]

The date the message was sent

See Also

Messages, MessageHistories

ZfLib.MessageHistory.Device

```
Property Device As String
```

The Device property returns the name of the device used to send the message.

Return Value

[out, retval]

The name of the device used

See Also

Messages, MessageHistories

ZfLib.MessageHistory.ErrorCode

```
Property ErrorCode As Long
```

The ErrorCode property returns the reason an attempt to send a message failed.

Return Value

[out, retval]

The returned Error code as a result of failure.

Remarks

Note that the values returned depend on the version of Zetafax Server running, not the version of API used. Newer version of the server will have additional error codes added.

See Also
ZfErr , Messages, MessageHistories

ZfLib.MessageHistory.Event

```
Property Event As EventEnum
```

The Event property returns the Event type of this MessageHistory object.

Return Value

[out, retval]

The retrieved EventEnum value.

See Also
Messages, Message, EventEnum

ZfLib.MessageHistory.Name

```
Property Name As String
```

The Name property returns the name of the recipient associated with this MessageHistory event

Return Value

[out, retval]

The name of the Remote server used

See Also

Messages, MessageHistories

ZfLib.MessageHistory.Organisation

```
Property Organisation As String
```

The Organisation property returns the organisation of the recipient associated with this MessageHistory event

Return Value

[out, retval]

The name of the Remote server used

See Also
Messages, MessageHistories

ZfLib.MessageHistory.PagesSent

```
Property PagesSent As Short
```

The PagesSent property returns the last page successfully sent.

Return Value

[out, retval]

The last page number sent

Remarks

If a three page message had several attempts at sending but only first two pages were sent successfully, this value would be set to 2.

See Also
Messages, MessageHistories

ZfLib.MessageHistory.RemoteServer

```
Property RemoteServer As String
```

The RemoteServer property returns the name of the remote server used to send a message when the message was sent using LCR.

Return Value

[out, retval]

The name of the Remote server used

See Also
Messages, MessageHistories

ZfLib.MessageHistory.Route

```
Property Route As RouteEnum
```

The Route property returns the route type used to send a message.

Return Value

[out, retval]

This is the route type used to send a message.

ZfLib.MessageHistory.RouteParams

```
Property RouteParams As String
```

The RouteParams property returns the route parameters used. For fax routes this is the fax number dialled.

Return Value

[out, retval]

The route parameters used to send a message.

See Also
Messages, MessageHistories

ZfLib.MessageInfo

The MessageInfo object contains information about a single Message object.

Groups

Read-only Properties

Body	The Body property returns the body name of the message's files.
Comment	The Comment property returns the comment associated with the message.
CustomField	The CustomField property returns the custom field data associated with the message.
ImageFilePath	
ImageSize	
ImageStream	
Organisation	The Organisation property returns the organisation of the first recipient to which this fax was sent. It is therefore only used in the Out directory.
Status	The Status property returns the status of a message.
Subject	The Subject property returns the subject of the

Type	message. The Type property returns the type of message you are looking at.
UserStatus	The UserStatus property returns the 'user status' of a message.

See Also
Message, MessageHistory

ZfLib.MessageInfo.Attachments

Property Attachments As ZfLib.Attachments

This property retrieves the Attachments collection.

Return Value

[out, retval]

The retrieved Attachments collection

Remarks

If ZfLib.Attachments.Count returns 0 there are no attachments associated with the fax.

ZfLib.MessageInfo.Body

```
Property Body As String
```

The Body property returns the body name of the message's files.

Return Value

[out, retval]

The message body name

See Also
Messages, Message

ZfLib.MessageInfo.Comment

```
Property Comment As String
```

The Comment property returns the comment associated with the message.

Return Value

[out, retval]

The Comment line of a message

ZfLib.MessageInfo.Covertext

Property CoverText As String

The CoverText property returns the covertext of the message

Return Value

[out, retval]

The covertext of a message

ZfLib.MessageInfo.CustomField

```
Property CustomField As String
```

The CustomField property returns the custom field data associated with the message.

ZfLib.MessageInfo.Files

Property Files As ZfLib.Files

This property retrieves the Files collection

Return Value

[out, retval]

The retrieved Files collection

Remarks

If ZfLib.Files.Count returns 0 no files were attached to the fax.

ZfLib.MessageInfo.ImageFilePath

```
Property ImageFilePath As String
```

ZfLib.MessageInfo.ImageSize

```
Property ImageSize As Long
```

ZfLib.MessageInfo.ImageStream

```
Property ImageStream As IStream
```

ZfLib.MessageInfo.Organisation

```
Property Organisation As String
```

The Organisation property returns the organisation of the first recipient to which this fax was sent. It is therefore only used in the Out directory.

Return Value

[out, retval]

The name of the Organisation to which this message was sent

ZfLib.MessageInfo.Status

```
Property Status As StatusEnum
```

The Status property returns the status of a message.

Return Value

[out, retval]

The status of a message.

ZfLib.MessageInfo.Subject

```
Property Subject As String
```

The Subject property returns the subject of the message.

Return Value

[out, retval]

The Subject line of a message

See Also
Messages, Message

ZfLib.MessageInfo.Type

```
Property Type As FaxTypeEnum
```

The Type property returns the type of message you are looking at.

Return Value

[out, retval]

The type of message at which you are looking

RemarksNote:This field defaults to type zfFaxTypeFax.

See Also
FaxTypeEnum, Message

ZfLib.MessageInfo.UserStatus

```
Property UserStatus As UserStatusEnum
```

The UserStatus property returns the 'user status' of a message.

Return Value

[out, retval]

The 'user status' of a message.

Remarks

The 'user satus' refers to the user's awareness of the message rather than the actual 'status' of the message ('read' or 'unread' rather than 'OK' or 'Failed').

ZfLib.Messages

The Messages collection contains Message objects from either a User's Inbox or Outbox .

Remarks
You do not add objects to the collection directly. To add an item to the Inbox you must send a fax, and add an item in the Outbox you must receive a fax.

This collection can contain thousands of objects.

Groups

Read-only Properties

Count	The Count property returns the number of Message items in the collection.
Item	The Item method retrieves a Message item from the collection.

MsgDir — The MsgDir property returns the full path of the directory which holds the messages in this collection.

ZfLib.Messages.Count

```
Property Count As Long
```

The Count property returns the number of Message items in the collection.

Return Value

[out, retval]

The number of Message items in the Messages collection.

See Also
Inbox , Outbox

ZfLib.Messages.Item

Property Item(ByVal var As Variant) As ZfLib .Message

The Item method retrieves a Message item from the collection.

Parameters

var

[in]

The index of the Message object we wish to retrieve from the collection

Return Value

The Message object retrieved from the Messages collection

See Also
Inbox , Outbox

ZfLib.Messages.MsgDir

```
Property MsgDir As String
```

The MsgDir property returns the full path of the directory which holds the messages in this collection.

Return Value

[out, retval]

The retrieved message directory

See Also
Inbox , Outbox , Messages

ZfLib.NewMessage

The NewMessage object allows the logged on user to prepare and submit a new message to the Zetafax server for sending.

Remarks

At least one Recipient must be configured before sending.
The message is sent using a submit file.

Groups

Operations

Send	The Send method submits the prepared message to the Zetafax server for sending.

Read-only Properties

Attachments	This property retrieves the Attachments collection used to add Zetafax attachments to the new message.
Body	After a message has been sent successfully this property contains the message body name. This is useful when you wish to keep track of the message in the Outbox .
Files	This property retrieves the Files collection.
Recipients	This property retrieves the Recipients collection.

#Read/Write Properties

After	The After property specifies the time and date after which the message is to be sent.
Charge	The Charge property specifies the billing code to use for a message.
Comment	The Comment property specifies the message description usually displayed on the right of the Zetafax client OUT window.
CoverSheet	The CoverSheet property specifies the Coversheet to be added to the new message.
Covertext	The Covertext property specifies the text to be added to the selected coversheet.
Delete	The Delete property specifies whether the server should delete the message from the OUT folder after it has been sent.
Format	This property specifies the file format of the data file to be sent.
From	The From property specifies the name that is put on the message Coversheet as the sender of the message.
Header	The Header property specifies the format of the Header line to appear at the top of each page of the message.
Hold	The Hold property specifies whether the message will be held once it is in the queue, i.e. it will not be sent until the user releases it.
Letterhead	The Letterhead property specifies a file (Letterhead) on which to merge the message before sending.
Preview	The Preview property specifies whether a preview file is to be prepared for the message.
Priority	This property specifies priority of the message.
Quality	The Quality property specifies the resolution to be used when sending the message.
SendTime	The SendTime property specifies when the message should be sent.
Subject	The Subject property specifies a message Subject line, which is displayed on the coversheet an in the Zetafax Client's out window.
Text	The Text property specifies the Text of the new message.
User	The User property specifies the Zetafax user submiting the message.
CustomField	The Custom Field property allows users of the API to specify their own custom data

See Also
Recipients , Attachments ,

ZfLib.NewMessage.After

```
Property After As Date
```

The After property specifies the time and date after which the message is to be sent.

Parameters

 after

[in]

The date and time to send a message.

Return Value

[out, retval]

The date and time to send the new message.

Remarks

This field is only used when the SendTime property is set to *zfSendTimeAfter* .

See Also
SendTime

ZfLib.NewMessage.Attachments

```
Property Attachments As ZfLib.Attachments
```

This property retrieves the Attachments collection used to add Zetafax attachments to the new message.

Return Value

[out, retval]

The retrieved Attachments collection.

Remarks

A new message does not need to contain any attachments.

See Also

Attachment , NewMessage , Files

ZfLib.NewMessage.Body

```
Property Body As String
```

After a message has been sent successfully this property contains the message body name. This is useful when you wish to keep track of the message in the Outbox .

Return Value

[out, retval]

The retrieved message body.

See Also
Outbox.GetMsg

ZfLib.NewMessage.Charge

```
Property Charge As String
```

The Charge property specifies the billing code to use for a message.

Parameters

 strCharge

[in]

Return Value

[out, retval]

The charge code used for a message.

Remarks

The charge codes are stored in the billing log when the fax completes, and may be used for charging the fax to a particular department/client.

ZfLib.NewMessage.Comment

```
Property Comment As String
```

The Comment property specifies the message description usually displayed on the right of the Zetafax client OUT window.

Parameters

strComment

[in]

Return Value

[out, retval]

The retrieved Comment line of the message.

Remarks

If the Comment line is omitted a message description detailing the first addressee is used.

ZfLib.NewMessage.CoverSheet

```
Property CoverSheet As String
```

The CoverSheet property specifies the Coversheet to be added to the new message.

Parameters

strCoverSheet

[in]

Return Value

[out, retval]

The CoverSheet used with the message.

Remarks

To specify the coversheet use the name by which it is known in Zetafax, not it's file name.

ZfLib.NewMessage.Covertext

```
Property Covertext As String
```

The Covertext property specifies the text to be added to the selected coversheet.

Parameters

strCovertext

[in]

Return Value

[out, retval]

The Covertext to be inserted onto the coversheet.

See Also
CoverSheet

ZfLib.NewMessage.CustomField

```
Property CustomField As String
```

The Custom Field property allows users of the API to specify their own custom data

ZfLib.NewMessage.Delete

```
Property Delete As Boolean
```

The Delete property specifies whether the server should delete the message from the OUT folder after it has been sent.

Parameters

fDelete

[in]

Return Value

[out, retval]

The retrieved Delete flag.

ZfLib.NewMessage.Files

```
Property Files As ZfLib.Files
```

This property retrieves the Files collection.

Return Value

[out, retval]

The retrieved Files collection.

Remarks

A new message does not need to contain any files.

See Also
File , NewMessage , Attachments

ZfLib.NewMessage.Format

```
Property Format As FormatEnum
```

This property specifies the file format of the data file to be sent.

Parameters

format

[in]

The Format type to use for this message

Return Value

[out, retval]

Remarks

In a standard submit file the format should be "ASCII" or "EPSON". Epson should be used if the message text contains any %% formatting commands, for

example

%%[BOLD: ON]

See Also
FormatEnum

ZfLib.NewMessage.From

```
Property From As String
```

The From property specifies the name that is put on the message Coversheet as the sender of the message.

Parameters

strFrom

[in]

Return Value

[out, retval]

The current From line of the new message

ZfLib.NewMessage.Header

```
Property Header As HeaderEnum
```

The Header property specifies the format of the Header line to appear at the top of each page of the message.

Parameters

header

[in]

The new Header type to use for this message

Return Value

[out, retval]

The current Header type of this message

Remarks

Specify more than one field OR the values together as for example:

zfHeaderNumber Or zfHeaderTo

See Also
HeaderEnum

ZfLib.NewMessage.Hold

Property Hold As Boolean

The Hold property specifies whether the message will be held once it is in the queue, i.e. it will not be sent until the user releases it.

Parameters

 fHold

[in]

Whether we want to hold this message.

Return Value

[out, retval]

Whether this message is to be held.

See Also

Message.Hold, Message.Release

ZfLib.NewMessage.Letterhead

Property

```
Letterhead As String
```

The Letterhead property specifies a file (Letterhead) on which to merge the message before sending.

Parameters

 strLetterhead
[in]

Return Value

[out, retval]
The current Letterhead to be used for this message.

Remarks
To specify the letterhead use the name by which it is known in Zetafax, not it's file name.

ZfLib.NewMessage.Preview

```
Property Preview As Boolean
```

The Preview property specifies whether a preview file is to be prepared for the message.

Return Value

[out, retval]

Whether we currently want a message preview.

ZfLib.NewMessage.Priority

```
Property Priority As PriorityEnum
```
This property specifies priority of the message.

Parameters

 priority

[in]

This is the new Priority of the message

Return Value

[out, retval]

The current Priority of the message

See Also
NewMessage , PriorityEnum

ZfLib.NewMessage.Quality

```
Property Quality As QualityEnum
```

The Quality property specifies the resolution to be used when sending the message.

Parameters

quality

[in]

The new resolution with which to send this message

Return Value

[out, retval]

The current resolution the message

See Also
NewMessage , QualityEnum

ZfLib.NewMessage.Recipients

```
Property Recipients As ZfLib.Recipients
```

This property retrieves the Recipients collection.

Return Value

[out, retval]

The retrieved Recipients collection.

Remarks

The recipients object must contain one or more recipients otherwise the sending of the message will fail.

See Also
NewMessage , Recipient

ZfLib.NewMessage.Send

```
Sub Send( )
```

The Send method submits the prepared message to the Zetafax server for sending.

Remarks

At least one recipient should be specified for this method to succeed.

ZfLib.NewMessage.SendTime

```
Property SendTime As SendTimeEnum
```

The SendTime property specifies when the message should be sent.

Parameters

sendTime

[in]

The new SendTime of the message.

Return Value

[out, retval]

The current SendTime of the message.

Remarks

If SendTime is set to *zfSendTimeAfter* then you need to specify the time to send using the After property.

See Also
After

ZfLib.NewMessage.Subject

```
Property Subject As String
```

The Subject property specifies a message Subject line, which is displayed on the coversheet an in the

Zetafax Client's out window.

Parameters

strSubject

[in]

Return Value

[out, retval]

The retrieved Subject line this message

Remarks

The maximum subject field is 60 characters long.

ZfLib.NewMessage.Text

```
Property Text As String
```

The Text property specifies the Text of the new message.

Parameters

strText

[in]

Return Value

[out, retval]

The retrieved body text of the new message.

ZfLib.NewMessage.User

```
Property User As String
```

The User property specifies the Zetafax user submiting the message.

Parameters

strUser

[in]

Return Value

[out, retval]

The current User submiting this message

See Also
UserSession

ZfLib.Outbox

The Outbox object represents a user's Zetafax OUT directory.

Remarks
This object is the same as the Inbox object except that it points to a different folder.

Groups
Operations

CheckNewMsgStatus	The CheckNewMsgStatus method checks whether the status of any of the messages in the current user's OUT directory have changed.
GetMsg	The GetMsg method returns the Message object matching the specified message body name.
GetMsgList	The GetMsgList method returns a Messages collection containing all the items in this directory

See Also
Inbox , Messages, Message

ZfLib.Outbox.CheckNewMsgStatus

```
Function CheckNewMsgStatus( ) As Boolean
```

The CheckNewMsgStatus method checks whether the status of any of the messages in the current user's OUT directory have changed.

Return Value

Returns True if there are new messages or a message's status has changed

See Also
ZfAPI , UserSession , Inbox

ZfLib.Outbox.GetMsg

```
Function GetMsg( ByVal bszMsg As String ) As ZfLib .Message
```

The GetMsg method returns the Message object matching the specified message body name.

Parameters

bszMsg

The body name of the message to retrieve

See Also
Messages, Message

ZfLib.Outbox.GetMsgList

```
Function GetMsgList( ) As ZfLib .Messages
```

The GetMsgList method returns a Messages collection containing all the items in this directory

Return Value

The retrieved Messages object.

See Also
Inbox , Messages

ZfLib.Recipient

The Recipient object contains address details of an addressee for whom a message is intended.

Remarks

Both the To and Fax properties need to be specified prior to sending a message.

If you use Recipients various Add methods you will not need to access this object directly.

Groups

Operations

Delete	The Delete method removes this recipient from the Recipients collection.

Read/Write Properties

Fax	The Fax property retrieves the Fax number of the recipient of a message.
Organisation	This property specifies the organisation to which the recipient belongs.
To	This property specifies the recipient's name.
Type	The Type property specifies how the message will be sent to the recipient.

See Also
NewMessage , Recipients.AddFaxRecipient , Recipients.AddLANRecipient

ZfLib.Recipient.Delete

```
Sub Delete( )
```

The Delete method removes this recipient from the Recipients collection.

See Also
Recipients , NewMessage

ZfLib.Recipient.Fax

```
Property Fax As String
```

The Fax property retrieves the Fax number of the recipient of a message.

Parameters

strFax

[in]

This is the new Fax number of the recipient.

Return Value

[out, retval]

The current Fax number of the recipient.

Remarks

This field could perhaps be better described as the address. If we were sending a LAN type address then the Fax property would actually contain the Zetafax user name of the person being sent the fax.

See Also
Recipients

ZfLib.Recipient.Organisation

```
Property Organisation As String
```

This property specifies the organisation to which the recipient belongs.

Parameters

 strOrganisation

[in]

The new Organistaion name of the recipient.

Return Value

[out, retval]

The current organisation name.

ZfLib.Recipient.To

```
Property To As String
```

This property specifies the recipient's name.

Parameters

 strRecipient

[in]

The name of the recipient of the new message.

Return Value

[out, retval]

The current name of the recipient.

ZfLib.Recipient.Type

```
Property Type As FaxTypeEnum
```

The Type property specifies how the message will be sent to the recipient.

Parameters

faxType

[in]

The new way the message will be sent.

Return Value

[out, retval]

The current way the message will be sent.

See Also
Recipients , FaxTypeEnum

ZfLib.Recipients

The Recipients collection object contains Recipient objects.

Remarks

This collection cannot exceed 30 objects.

New objects are added using the various AddRecipient methods which create a properly configured Recipient object. This means you will rarely have to access a recipient object directly.

Groups

Operations

AddFaxRecipient	The AddFaxRecipient method adds a Fax Recipient item to the collection.
AddLANRecipient	The AddLANRecipient method adds a LAN Recipient item to the collection.
AddSMSRecipient	The AddSMSRecipient method adds an SMS Recipient item to the collection.

Read-only Properties

Count	The Count method returns the number of Recipient items in the collection.
Item	The Item property retrieves a Recipient item from the collection.

See Also
Recipient

equisys ZETAfax

ZfLib.Recipients.AddFaxRecipient

```
Function AddFaxRecipient( ByVal bszTo As String, ByVal bszOrganisation As String, ByVal bszFax As String ) As ZfLib.Recipient
```

The AddFaxRecipient method adds a Fax Recipient item to the collection.

Parameters

bszTo

The name of the recipient.

bszOrganisation

The name of the organnisation to which the recipient belongs.

bszFax

The recipient's Fax Number.

Return Value

The new Recipient object.

See Also
NewMessage , FaxTypeEnum

ZfLib.Recipients.AddLANRecipient

```
Function AddLANRecipient( ByVal bszTo As String, ByVal bszOrganisation As String, ByVal bszUser As String ) As ZfLib.Recipient
```

The AddLANRecipient method adds a LAN Recipient item to the collection.

Parameters

bszTo
The name of the recipient.

bszOrganisation
The name of the organnisation which the recipient belongs.

bszUser
The LAN Address of the recipient.

Return Value

The new Recipient object.

See Also
NewMessage , Recipients.AddFaxRecipient , FaxTypeEnum

ZfLib.Recipients.AddSMSRecipient

```
Function AddSMSRecipient( ByVal bszTo As String, ByVal bszOrganisation As String, ByVal bszSMS As String ) As ZfLib.Recipient
```

The AddSMSRecipient method adds an SMS Recipient item to the collection.

Parameters

bszTo
The name of the Recipient .

bszOrganisation
The organisation to which the recipient belongs.

bszSMS
The SMS number of the recipient.

Return Value

The new Recipient object.

See Also
NewMessage , Recipients.AddFaxRecipient , FaxTypeEnum

ZfLib.Recipients.Count

```
Property Count As Long
```

The Count method returns the number of Recipient items in the collection.

Return Value

[out, retval]

The number of items in the Recipients collection

ZfLib.Recipients.Item

```
Property Item( ByVal var As Variant ) As ZfLib.Recipient
```

The Item property retrieves a Recipient item from the collection.

Parameters

 var

[in]

The index of the item to retrieve from the collection.

Return Value

The Recipient object retrieved from the Recipients collection.

ZfLib.Server

The Server object allows control over the Zetafax server, and access to objects describing the server's state.

Groups

Operations

Check	This is a method to check if the Zetafax server is running.
GetServerInfo	The GetServerInfo method returns the ServerInfo object which contains information about the state of the Zetafax server.
Restart	This method restarts the Zetafax Server.
Start	This method starts the Zetafax Server if it is stopped.
Stop	This method stops the Zetafax Server if it is running.

See Also
ServerInfo

ZfLib.Server.Check

```
Function Check( ) As Boolean
```

This is a method to check if the Zetafax server is running.

Return Value

If the server is running this method returns True.

ZfLib.Server.GetServerInfo

```
Function GetServerInfo( ) As ZfLib.ServerInfo
```

The GetServerInfo method returns the ServerInfo object which contains information about the state of the Zetafax server.

Return Value

This method returns the ServerInfo object.

ZfLib.Server.Restart

```
Sub Restart( )
```

This method restarts the Zetafax Server.

ZfLib.Server.Start

```
Sub Start( )
```

This method starts the Zetafax Server if it is stopped.

ZfLib.Server.Stop

```
Sub Stop( )
```

This method stops the Zetafax Server if it is running.

ZfLib.ServerInfo

The ServerInfo object exposes the status and configuration of the Zetafax server.

Groups

Read-only Properties

Coversheets	The Coversheets property returns a Coversheets collection with information about the Coversheets on the Zetafax server.
Deferred	The deferred property retrieves the number of items in the queue that have been deferred.
Devices	The Devices property returns a Devices collection detailing the current state of the Zetafax server's devices.
Letterheads	The Letterheads property returns a Letterheads collection with information about the Letterheads on the Zetafax server.
Links	The Links property returns a Links collection with information about the

	current state of the LCR links maintianed by the Zetafax server.
MaxDevices	The MaxDevices property retrieves the maximum number of devices the Zetafax server supports.
MaxLinks	The MaxLinks property retrieves the maximum number of LCR links the Zetafax server supports.
RemoteAccept	The RemoteAccept property retrieves the number of message items currently accepted for sending by remote servers.
RouterSub	The RouterSub property retrieves the number of items submitted to the Router for sending and awaiting acceptance by a remote server.
Scanning	The Scanning property retrieves the number of items in the queue being processed by a scanning device.
Sending	The Sending property returns the number of items in the queue which are currently being sent.
WaitingConvert	The WaitingConvert property retrieves the number of items in the queue waiting to be converted or being prepared for sending.
WaitingDevice	The WaitingDevice property retrieves the number of items waiting for a device to become available.
WaitingResend	The WaitingResend property retrieves the number of items in the queue waiting before retrying after a send attempt failed.

ZfLib.ServerInfo.Coversheets

```
Property Coversheets As ZfLib.Coversheets
```

The Coversheets property returns a Coversheets collection with information about the Coversheets on the Zetafax server.

Return Value

[out, retval]

The Coversheets collection.

See Also
Coversheets, Coversheet

ZfLib.ServerInfo.Deferred

```
Property Deferred As Short
```

The deferred property retrieves the number of items in the queue that have been deferred.

Return Value

[out, retval]

The number of deferred message items.

ZfLib.ServerInfo.Devices

```
Property Devices As ZfLib.Devices
```

The Devices property returns a Devices collection detailing the current state of the Zetafax server's devices.

Return Value

[out, retval]

The Devices collection.

See Also
Devices

ZfLib.ServerInfo.Letterheads

```
Property Letterheads As ZfLib.Letterheads
```

The Letterheads property returns a Letterheads collection with information about the Letterheads on the Zetafax server.

Return Value

[out, retval]

The Letterheads collection.

See Also
Letterheads, Letterhead

ZfLib.ServerInfo.Links

```
Property Links As ZfLib.Links
```

The Links property returns a Links collection with information about the current state of the LCR links maintianed by the Zetafax server.

Return Value

[out, retval]

The Links collection.

See Also
Links

ZfLib.ServerInfo.MaxDevices

```
Property MaxDevices As Short
```

The MaxDevices property retrieves the maximum number of devices the Zetafax server supports.

Return Value

[out, retval]

The maximum number of devices allowed

Remarks

This property always returns 100.

ZfLib.ServerInfo.MaxLinks

```
Property MaxLinks As Short
```

The MaxLinks property retrieves the maximum number of LCR links the Zetafax server supports.

Return Value

[out, retval]

The maximum number of Remote Server links allowed

ZfLib.ServerInfo.RemoteAccept

```
Property RemoteAccept As Short
```

The RemoteAccept property retrieves the number of message items currently accepted for sending by remote servers.

Return Value

[out, retval]

The number of items accepted by remote server.

ZfLib.ServerInfo.RouterSub

```
Property RouterSub As Short
```

The RouterSub property retrieves the number of items submitted to the Router for sending and awaiting acceptance by a remote server.

Return Value

[out, retval]

This is the number of items submitted to Router

ZfLib.ServerInfo.Scanning

```
Property Scanning As Short
```

The Scanning property retrieves the number of items in the queue being processed by a scanning device.

Return Value

[out, retval]

The numbers of scan request items in queue.

ZfLib.ServerInfo.Sending

```
Property Sending As Short
```

The Sending property returns the number of items in the queue which are currently being sent.

Return Value

[out, retval]

The number of message items in the queue being sent.

ZfLib.ServerInfo.WaitingConvert

```
Property WaitingConvert As Short
```

The WaitingConvert property retrieves the number of items in the queue waiting to be converted or being prepared for sending.

Return Value

[out, retval]

This is the number of items waiting to be converted.

ZfLib.ServerInfo.WaitingDevice

```
Property WaitingDevice As Short
```

The WaitingDevice property retrieves the number of items waiting for a device to become available.

Return Value

[out, retval]

The number of items awaiting device.

ZfLib.ServerInfo.WaitingResend

```
Property WaitingResend As Short
```

The WaitingResend property retrieves the number of items in the queue waiting before retrying after a send attempt failed.

Return Value

[out, retval]

The number of message items awaiting resend.

ZfLib.UserSession

The UserSession object contains details about a Zetafax user's configuration, messages and gives you the ability to create new messages for sending.

Groups

Operations

CreateNewMsg	The CreateNewMsg method creates a NewMessage object.
Logoff	Logs of the user logged on in ZfAPI.Logon
SendSubmitFile	This method interpretes the given SUBMIT format file, creating a CONTROL file and a DATA file in ASCII text or EPSON print format. It then submits these files for sending.

Read-only Properties

APIPrint	The APIPrint property returns the APIPrint object
Coversheet	The Coversheet property returns the user's default Coversheet .
FromName	The FromName property specifies the name of the logged on user as it appears in the from field of new messages.
Inbox	The Inbox property returns the Inbox object.
Outbox	The Outbox property returns the Outbox object

Server	The Server property returns the Server object.
SystemArea	The SystemArea property returns the Zetafax System directory.
UserArea	The UserArea property returns path to the logged on Zetafax user's directory.
UserInDir	The UserInDir property returns the User's IN directory.
UserOutDir	The UserOutDir property returns the User's OUT directory.

See Also
NewMessage , Inbox , Outbox

ZfLib.UserSession.APIPrint

```
Property APIPrint As ZfLib.APIPrint
```

The APIPrint property returns the APIPrint object

Return Value

[out, retval]

An APIPrint object.

See Also
APIPrint

ZfLib.UserSession.Coversheet

```
Property Coversheet As String
```

The Coversheet property returns the user's default Coversheet.

Return Value

[out, retval]

The user's default Coversheet.

ZfLib.UserSession.CreateNewMsg

```
Function CreateNewMsg( ) As ZfLib.NewMessage
```

The CreateNewMsg method creates a NewMessage object.

Return Value

The new NewMessage object.

See Also
NewMessage

ZfLib.UserSession.FromName

```
Property FromName As String
```

The FromName property specifies the name of the logged on user as it appears in the from field of new messages.

Return Value

[out, retval]

The user's FromName.

ZfLib.UserSession.Inbox

Property Inbox As ZfLib.Inbox

The Inbox property returns the Inbox object.

Return Value

[out, retval]

An Inbox object.

See Also
Inbox

ZfLib.UserSession.Logoff

```
Sub Logoff( )
```

Logs of the user logged on in ZfAPI.Logon

Remarks

Note:This function will also be called during the object's destruction. It is primarily intended for occassions when you don't know when the object will be released.

ZfLib.UserSession.Outbox

```
Property Outbox As ZfLib.Outbox
```

The Outbox property returns the Outbox object

Return Value

[out, retval]

A Outbox object.

See Also
Outbox

ZfLib.UserSession.SendSubmitFile

```
Function SendSubmitFile( ByVal bszSubFile As String, ByVal bszPrefix As String ) As String
```

This method interpretes the given SUBMIT format file, creating a CONTROL file and a DATA file in ASCII text or EPSON print format. It then submits these files for sending.

Parameters

bszSubFile
The name of the submit file

bszPrefix
Prefix to use when creating control and data files

Return Value

The name of message body if successfully submitted.

See Also
NewMessage

ZfLib.UserSession.Server

```
Property Server As ZfLib .Server
```

The Server property returns the Server object.

Return Value

[out, retval]

The Server object.

See Also
Server

ZfLib.UserSession.SystemArea

```
Property SystemArea As String
```

The SystemArea property returns the Zetafax System directory.

Return Value

[out, retval]

The path of the Zetafax system directory.

ZfLib.UserSession.UserArea

```
Property UserArea As String
```

The UserArea property returns path to the logged on Zetafax user's directory.

Return Value

[out, retval]

The path to the User's area

ZfLib.UserSession.UserInDir

```
Property UserInDir As String
```

The UserInDir property returns the User's IN directory.

Return Value

[out, retval]

The path of the User's IN directory.

ZfLib.UserSession.UserOutDir

```
Property UserOutDir As String
```

The UserOutDir property returns the User's OUT directory.

Return Value

[out, retval]

The path to the User's OUT directory.

ZfLib.ZfAPI

The ZfAPI object is the Application object of the COM version of the API, and is the only object that can be created directly. You must create this object and logon before you can access any of the other objects.

Groups

Operations

GetZetafaxServerInfoFromAD	Gets Zetafax directories configured for the specified Zetafax server.
GetZetafaxServersFromAD	Returns a list of Zetafax servers registered in Active Directory.
Logon	The Logon method logs on a user and returns the UserSession object.
LogonAnonymous	The LogonAnonymous method returns a limited UserSession object that is not logged on as a specific user.
SetZetafaxDirs	Explicitly state the locations of the Zetafax directories instead of letting the API read zfclient.ini.
SetZetafaxServerFromAD	Explicitly state the Zetafax server to use instead of letting the API read zfclient.ini.

Read-only Properties

RequestDir	Gets the path to the Request directory set by SetZetafaxDirs or SetZetafaxServerFromAD .
ServerDir	Gets the path to the Server directory set by SetZetafaxDirs or SetZetafaxServerFromAD .
SystemDir	Gets the path to the System directory set by SetZetafaxDirs or SetZetafaxServerFromAD .
UsersDir	Gets the path to the Users directory set by SetZetafaxDirs or SetZetafaxServerFromAD .
Version	The Version property returns the current version of ZfLib library
ZetafaxServer	Gets the server set by SetZetafaxServerFromAD

ZfLib.ZfAPI.GetZetafaxServerInfoFromAD

```
Sub GetZetafaxServerInfoFromAD( ByVal bszServerName As String, ByRef pbszServerDir As
String, ByRef pbszSystemDir As String, ByRef pbszUsersDir As String, ByRef p
bszRequestDir As String )
```

Gets Zetafax directories configured for the specified Zetafax server.

Parameters

bszServerName
The server name to look up in AD

pbszServerDir
The returned Server directory

pbszSystemDir
The returned System directory

pbszUsersDir
The returned Users directory

 pbszRequestDir
The returned Request directory

Return Value

If the directories have not been set by the Share Wizard, the method will return zfErrE_ACTIVE_DIRECTORY_MISSING_VALUE.

Remarks

This method queries Active Directory for the Zetafax server specified by bszServerName, and gets the directories that have been configured for the server by the Share Wizard.

See Also
ZfAPI.GetZetafaxServersFromAD

ZfLib.ZfAPI.GetZetafaxServersFromAD

```
Function GetZetafaxServersFromAD( ) As Variant
```

Returns a list of Zetafax servers registered in Active Directory.

Return Value

The returned server list

Remarks

This method queries Active Directory for installed Zetafax servers. It searches the Global Catalog if available, so will find any Zetafax servers installed in
the forest.

See Also
ZfAPI.SetZetafaxDirs , ZfAPI.GetZetafaxServerInfoFromAD

ZfLib.ZfAPI.Logon

Function Logon(ByVal bszUserName As String, ByVal fExclusive As Boolean) As ZfLib.UserSession

The Logon method logs on a user and returns the UserSession object.

Parameters

 bszUserName
The Zetafax User account

 fExclusive

Exclusive Logon flag

Return Value

The returned UserSession object

Remarks

A successful call to this method is required to retrieve a UserSession object which allows access to all the other objects in ZfLib library. The method may be called more than once by the program if it wishes to logon more than one user. The fExclusive Boolean flag is set to FALSE if the specified user is permitted to be logged on elsewhere (either on a normal Zetafax client, or in another API program).

See Also
ZfAPI.LogonAnonymous , UserSession , UserSession.Logoff

ZfLib.ZfAPI.LogonAnonymous

```
Function LogonAnonymous( ) As ZfLib.UserSession
```

The LogonAnonymous method returns a limited UserSession object that is not logged on as a specific user.

Return Value

The returned UserSession object
Remarks
An anonymous session is restricted to the following methods in the UserSession object:
UserSession.Server
UserSession.SystemArea
UserSession.Logoff
All other methods return an error.
See Also
ZfAPI.Logon , UserSession , UserSession.Logoff

ZfLib.ZfAPI.RequestDir

```
Property RequestDir As String
```

Gets the path to the Request directory set by SetZetafaxDirs or SetZetafaxServerFromAD .

Return Value

[out, retval]

The returned Request directory

ZfLib.ZfAPI.ServerDir

```
Property ServerDir As String
```

Gets the path to the Server directory set by SetZetafaxDirs or SetZetafaxServerFromAD .

Return Value

[out, retval]

The returned Server directory

ZfLib.ZfAPI.SetZetafaxDirs

```
Sub SetZetafaxDirs( ByVal bszServerDir As String, ByVal bszSystemDir As String, ByVal
bszUsersDir As String, ByVal bszRequestDir As String )
```

Explicitly state the locations of the Zetafax directories instead of letting the API read zfclient.ini.

Return Value

Returns E_FAIL if a session has already been created on this object.

Remarks

The API normally requires the Zetafax client to be installed and uses the zfclient.ini file to find the Zetafax server. In some cases the client may not be used so this method allows the caller to specify the location of the Zetafax server.

See Also
ZfAPI.SetZetafaxServerFromAD

ZfLib.ZfAPI.SetZetafaxServerFromAD

```
Sub SetZetafaxServerFromAD( ByVal bszServerName As String )
```

Explicitly state the Zetafax server to use instead of letting the API read zfclient.ini.

Parameters

bszServerName

Name of Zetafax server machine

Return Value

Returns E_FAIL if a session has already been created on this object.

Remarks

The API normally requires the Zetafax client to be installed and uses the zfclient.ini file to find the Zetafax server. In some cases the client may not be used so this method allows the caller to specify the Zetafax server. This method looks up the Zetafax server in Active Directory, and sets the Zetafax server directories on this object.

See Also
ZfAPI.SetZetafaxDirs

ZfLib.ZfAPI.SystemDir

```
Property SystemDir As String
```

Gets the path to the System directory set by SetZetafaxDirs or SetZetafaxServerFromAD .

Return Value

[out, retval]

The returned System directory

ZfLib.ZfAPI.UsersDir

```
Property UsersDir As String
```

Gets the path to the Users directory set by SetZetafaxDirs or SetZetafaxServerFromAD .

Return Value

[out, retval]

The returned Users directory

ZfLib.ZfAPI.Version

```
Property Version As String
```

The Version property returns the current version of ZfLib library

Return Value

[out, retval]

The current ZfLib Version.

Remarks

API version

ZfLib.ZfAPI.ZetafaxServer

```
Property ZetafaxServer As String
```

Gets the server set by SetZetafaxServerFromAD

Return Value

[out, retval]

Returned server name

Remarks

Gets the server set by SetZetafaxServerFromAD

ZfLib

This is a list of the objects and enumerations that appear in the Zetafax COM library

Groups

COM Classes

API Print	This object allows the API to print via the Zetafax Printer.

Attachment	The Attachment object contains the name of a Zetafax attachment to be attached to a message before it is sent.
Attachments	The Attachments collection object contains Attachment objects.
Coversheet	The Coversheet class represents a Zetafax coversheet.
Coversheets	The Coversheets collection object contains Coversheet objects.
Device	The Device object contains the configuration information and status of a device attached to the Zetafax server.
Devices	The Devices collection contains Device objects.
File	The File object contains the path of a file to be attached to the message before it is sent.
Files	The Files collection object contains File objects. To attach Zetafax attachments to the message use the Attachments collection.
Inbox	The Inbox object represents a user's Zetafax IN directory.
Letterhead	The Letterhead class represents a Zetafax letterhead.
Letterheads	The Letterheads collection object contains Letterhead objects.
Link	The Link object allows a user to retrieve information on the status of a link. It also provides statistics on the Link 's performance.
Links	The Links collection contains Link objects.
Message	The Message object allows access to the details of sent and received messages.
MessageHistories	The MsgHistories collection represents a message's transmission history and contains MessageHistory objects.
MessageHistory	The MessageHistory object contains an item in a message's transmission history.
MessageInfo	The MessageInfo object contains information about a single Message object
Messages	The Messages collection contains Message objects from either a User's Inbox or Outbox .
NewMessage	The NewMessage object allows the logged on user to prepare and submit a new

	message to the Zetafax server for sending.
Outbox	The Outbox object represents a user's Zetafax OUT directory.
Recipient	The Recipient object contains address details of an addressee for whom a message is intended.
Recipients	The Recipients collection object contains Recipient objects.
Server	The Server object allows control over the Zetafax server, and access to objects describing the server's state.
ServerInfo	The ServerInfo object exposes the status and configuration of the Zetafax server.
UserSession	The UserSession object contains details about a Zetafax user's configuration, messages and gives you the ability to create new messages for sending.
ZfAPI	The ZfAPI object is the Application object of the COM version of the API, and is the only object that can be created directly. You must create this object and logon before you can access any of the other objects.

Type Definitions

DevStatusEnum	The DevStatusEnum enumeration specifies the current status of a Device.
EventEnum	The EventEnum enumeration specifies the event described in a MessageHistory object
FaxTypeEnum	The FaxTypeEnum enumeration specifies how a recipient will be sent a message
FormatEnum	The FormatEnum enumeration specifies the format of the data file to be sent
HeaderEnum	This HeaderEnum Enumeration specifies the content of the header line to appear in the top page of each fax
LinkStatusEnum	The LinkStatusEnum enumeration specifies the current status of an LCR link
PriorityEnum	The PriorityEnum enumeration specifies the priority of an outgoing message
QualityEnum	The QualityEnum enumeration specifies the resolution when sending a fax

RouteEnum	The RouteEnum enumeration specifies the type of route used
SendTimeEnum	The SendTimeEnum enumeration specifies when a message will be sent
StatusEnum	The StatusEnum specifies the current status of a message
UserStatusEnum	The StatusEnum specifies the 'user status' of a message (that is the user's awareness of the message)
ZfErr	The ZfErr enumeration specifies the errors returned the API.

ZfLib.DevStatusEnum

```
Enum ZfLib.DevStatusEnum

  zfDevStatusIdle  = 1
  zfDevStatusError = 2
  zfDevStatusFailed= 3
  zfDevStatusOffline  = 4
  zfDevStatusBusy  = 5
  zfDevStatusWaitingConnect = 6
  zfDevStatusSending  = 7
  zfDevStatusReceiving= 8
  zfDevStatusWaitingScanDoc = 11
  zfDevStatusIncomingCall= 13

End Enum
```

The DevStatusEnum enumeration specifies the current status of a Device.

Members

zfDevStatusBusy
The device is busy processing a message

zfDevStatusError
Trying to recover from an error

zfDevStatusFailed
Unable to recover from error

zfDevStatusIdle
The device is idle

zfDevStatusIncomingCall
The device has receive a request to recieve data

zfDevStatusOffline
The device has been turned off

zfDevStatusReceiving

Recieving data

zfDevStatusSending
Sending data

zfDevStatusWaitingConnect
The device is attempting to connect to a remote device

zfDevStatusWaitingScanDoc
Waiting for document to scan

See Also
Device

ZfLib.EventEnum

```
Enum ZfLib.EventEnum

  zfEventSentOK = 1
  zfEventRecdOK = 2
  zfEventScanOK = 3
  zfEventSentError = 4
  zfEventRecdError = 5
  zfEventScanError = 6
  zfEventTried  = 7
  zfEventRouterSub = 8
  zfEventRouterAcc = 9
  zfEventRouterErr = 10

End Enum
```

The EventEnum enumeration specifies the event described in a MessageHistory object

Members

zfEventRecdError
Received with errors

zfEventRecdOK
Received successfully

zfEventRouterAcc
Accepted for transmission by remote server

zfEventRouterErr
Submission to a remote server failed

zfEventRouterSub
Passed to remote Zetafax server for sending

zfEventScanError
Scanning request completed with errors

zfEventScanOK
Scanned successfully

zfEventSentError
Sent with errors

zfEventSentOK
Sent successfully

zfEventTried
Send attempt was unsuccessful

See Also
MessageHistory

ZfLib.FaxTypeEnum

```
Enum ZfLib.FaxTypeEnum

  zfFaxTypeFax = 1
  zfFaxTypeLAN = 2
  zfFaxTypeSMS = 3

End Enum
```

The FaxTypeEnum enumeration specifies how a recipient will be sent a message

Members

zfFaxTypeFax
Via fax

zfFaxTypeLAN
Via LAN (used to send messages to another Zetafax User)

zfFaxTypeSMS
Via SMS

See Also
Recipient

ZfLib.FormatEnum

```
Enum ZfLib.FormatEnum

  zfFormatASCII= 1
  zfFormatEpson= 2
  zfFormatTIFFNormal = 3
  zfFormatTIFFFine= 4

End Enum
```

The FormatEnum enumeration specifies the format of the data file to be sent

Members

zfFormatASCII
ASCII text (TXT)

zfFormatEpson
Epson FX or LQ print spool file (EPN)

zfFormatTIFFFine
200x200 dpi TIFF fax file (G3F)

zfFormatTIFFNormal
200x100 dpi TIFF fax file (G3N)

See Also
NewMessage

ZfLib.HeaderEnum

```
Enum ZfLib.HeaderEnum

  zfHeaderNone= 0
  zfHeaderNumber = 1
  zfHeaderTo  = 2
  zfHeaderFrom= 4
  zfHeaderDate= 8
  zfHeaderTime= 16

End Enum
```

This HeaderEnum Enumeration specifies the content of the header line to appear in the top page of each fax

Members

zfHeaderDate
Show date

zfHeaderFrom
Show name of sender

zfHeaderNone
Turn all header information off

zfHeaderNumber
Show page numbers

zfHeaderTime
Show time

zfHeaderTo
Show name of recipient

Remarks
The members of this enumeration can be ORed together

See Also
NewMessage.Header

ZfLib.LinkStatusEnum

```
Enum ZfLib.LinkStatusEnum
  zfLinkOnline = 1
  zfLinkOffline= 2
  zfLinkFailedWDog= 3
  zfLinkInitialising = 4
  zfLinkShutdown  = 5
  zfLinkUnknownState = 6
End Enum
```

The LinkStatusEnum enumeration specifies the current status of an LCR link

Members

zfLinkFailedWDog
The server failed to receive an expected status message

zfLinkInitialising
The link is initialising but has started communicating with the remote server

zfLinkOffline
The remote server has closed the link or the local link is temporarily closed

zfLinkOnline
The link functioning normally

zfLinkShutdown
The link has been permanently closed.

zfLinkUnknownState
The link is initialising and has not yet heard from the remote server

See Also
Link

ZfLib.PriorityEnum

```
Enum ZfLib.PriorityEnum
  zfPriorityBackground = 1
  zfPriorityNormal  = 2
  zfPriorityUrgent  = 3
```

```
End Enum
```

The PriorityEnum enumeration specifies the priority of an outgoing message

Members

zfPriorityBackground
Send at low priority

zfPriorityNormal
Send with normal priority

zfPriorityUrgent
Send urgently

See Also
NewMessage

ZfLib.QualityEnum

```
Enum ZfLib.QualityEnum
  zfQualityHigh= 1
  zfQualityNormal = 2
  zfQualityDraft  = 3
End Enum
```

The QualityEnum enumeration specifies the resolution when sending a fax

Members

zfQualityDraft
Standard quality (200x100 dpi)

zfQualityHigh
Fine quality (200x200 dpi)

zfQualityNormal
Send the fax at default quality (configured by the Zetafax administrator)

See Also
NewMessage

ZfLib.RouteEnum

```
Enum ZfLib.RouteEnum
  zfRouteFaxNormal = 2
  zfRouteFaxFine= 3
```

```
  zfRouteFaxSuperfine = 4
End Enum
```

The RouteEnum enumeration specifies the type of route used

Members

zfRouteFaxFine
Fine resolution fax (200x200 dpi)

zfRouteFaxNormal
Standard resolution fax (200x100 dpi)

zfRouteFaxSuperfine
Super-fine resolution fax (200x400 dpi)

See Also
MessageHistory

ZfLib.SendTimeEnum

```
Enum ZfLib.SendTimeEnum
  zfSendTimeNow  = 1
  zfSendTimeOffpeak = 2
  zfSendTimeAfter= 3
End Enum
```

The SendTimeEnum enumeration specifies when a message will be sent

Members

zfSendTimeAfter
Send after specified time

zfSendTimeNow
Send as soon as possible

zfSendTimeOffpeak
Send during offpeak rates

See Also
NewMessage

ZfLib.StatusEnum

```
Enum ZfLib.StatusEnum
  zfStatusAdding  = 1
```

```
  zfStatusHeld = 2
  zfStatusDeferred= 3
  zfStatusWaiting = 4
  zfStatusConverting = 6
  zfStatusConnecting = 7
  zfStatusSending = 8
  zfStatusAborting= 10
  zfStatusIncoming= 11
  zfStatusSubRouter  = 13
  zfStatusAcceptRemote  = 14
  zfStatusOk= 30
  zfStatusFailed  = 31
  zfStatusPreviewOK  = 32
  zfStatusPreviewFailed = 33
End Enum
```

The StatusEnum specifies the current status of a message

Members

zfStatusAborting
The message is currently being aborted

zfStatusAcceptRemote
Message accepted for transmission by remote server

zfStatusAdding
Message added to the queue

zfStatusConnecting
Connecting to remote device

zfStatusConverting
Being prepared for sending

zfStatusDeferred
Waiting for sending after a specific time

zfStatusFailed
Completed with one or more errors

zfStatusHeld
Message held by user

zfStatusIncoming
Being received by a device

zfStatusOk
Completed with no errors

zfStatusPreviewFailed
Failed to prepare for preview

zfStatusPreviewOK
Message ready for preview

zfStatusSending
Sending to remote device

zfStatusSubRouter
Passed to a remote Zetafax server for sending, but not yet acknowledged

zfStatusWaiting

Waiting for conversion or a free device to send the message

See Also
MessageInfo

ZfLib.UserStatusEnum

```
Enum ZfLib.UserStatusEnum
  zfUserStatusOK= 1
  zfUserStatusAware= 2
  zfUserStatusWaiting = 3
End Enum
```

The StatusEnum specifies the 'user status' of a message (that is the user's awareness of the message)

Members

zfUserStatusAware
The user is aware of message, but has not read it.

zfUserStatusOK
The message has been read.

zfUserStatusWaiting
Waiting for user to do something.

See Also
MessageInfo

ZfErr

```
Enum ZfErr
  zfErrE_INVALID_PARAM= &H8004F700
  zfErrE_NOT_INIT  = &H8004F701
  zfErrE_INIT_FAIL = &H8004F702
  zfErrE_INVALID_ZF_INIT_FILE  = &H8004F703
  zfErrE_INVALID_ZF_USER = &H8004F704
  zfErrE_CANNOT_LOG_ON= &H8004F705
  zfErrE_CANT_LOG_ON  = zfErrE_CANNOT_LOG_ON
  zfErrE_PATH_NOT_FOUND  = &H8004F706
  zfErrE_TOO_MANY_FILES  = &H8004F707
  zfErrE_FILE_CREATE_ERROR  = &H8004F708
  zfErrE_FILE_OPEN_ERROR = &H8004F709
  zfErrE_FILE_ERROR= &H8004F70A
  zfErrE_FILE_NOT_FOUND  = &H8004F70B
  zfErrE_SERVER_NOT_RUNNING = &H8004F70C
  zfErrE_INVALID_DATA_FORMAT= &H8004F70D
```

```
  zfErrE_MSG_UNKNOWN  = &H8004F70E
  zfErrE_MSG_NOT_COMPLETED  = &H8004F70F
  zfErrE_MSG_EXISTS= &H8004F710
  zfErrE_INFO_FILE_OPEN_ERROR  = &H8004F711
  zfErrE_INFO_FILE_ERROR = &H8004F712
  zfErrE_INFO_FILE_INVALID  = &H8004F713
  zfErrE_CANNOT_SUBMIT= &H8004F714
  zfErrE_CANT_SUBMIT_REQUEST= zfErrE_CANNOT_SUBMIT
  zfErrE_BUFFER_TOO_SMALL= &H8004F715
  zfErrE_SUBMIT_FILE_INVALID= &H8004F716
  zfErrE_CANNOT_READ_MSG_DEFAULTS = &H8004F717
  zfErrE_CANT_READ_MSG_DEFAULTS= zfErrE_CANNOT_READ_MSG_DEFAULTS
  zfErrE_CONTROL_FILE_OPEN_ERROR  = &H8004F718
  zfErrE_CONTROL_FILE_ERROR = &H8004F719
  zfErrE_CONTROL_FILE_INVALID  = &H8004F71A
  zfErrE_SERVER_RUNNING  = &H8004F71B
  zfErrE_NO_START_SYSMAN = &H8004F71C
  zfErrE_SERVER_INI_FILE_ERROR = &H8004F71E
  zfErrE_FUNCTION_ABORT  = &H8004F71F
  zfErrE_TIMEOUT_EXPIRED = &H8004F720
  zfErrE_MALLOC_FAILED= &H8004F724
  zfErrE_LICENCE_ERROR= &H8004F725
  zfErrE_API_LICENSE_ERROR  = &H8004F726
  zfErrE_USER_NOT_INIT= &H8004F727
  zfErrE_ACCESSDENIED = &H8004F728
  zfErrE_REGISTRY_ERROR  = &H8004F729
  zfErrE_ACTIVE_DIRECTORY_NOT_PRESENT= &H8004F72A
  zfErrE_ACTIVE_DIRECTORY_ERROR= &H8004F72B
  zfErrE_ACTIVE_DIRECTORY_MISSING_VALUE = &H8004F72C
  zfErrE_FULLCOLLECTION  = &H8004F730
End Enum
```

The ZfErr enumeration specifies the errors returned the API.

Members

zfErrE_ACCESSDENIED
Access Denied

zfErrE_ACTIVE_DIRECTORY_ERROR
Active Directory error

zfErrE_ACTIVE_DIRECTORY_MISSING_VALUE
Active Directory value not set

zfErrE_ACTIVE_DIRECTORY_NOT_PRESENT
Active Directory server not present

zfErrE_API_LICENSE_ERROR
Your Zetafax licence does not permit you to use the API

zfErrE_BUFFER_TOO_SMALL
The buffer in which to return data was too small

zfErrE_CANNOT_LOG_ON
The user was already logged on or the API cannot access their directories

zfErrE_CANNOT_READ_MSG_DEFAULTS
Cannot read user's USER.INI

zfErrE_CANNOT_SUBMIT
Error handling .SUB file

zfErrE_CONTROL_FILE_ERROR
Error reading/writing CTL file

zfErrE_CONTROL_FILE_INVALID
CTL file is corrupt

zfErrE_CONTROL_FILE_OPEN_ERROR
The API was unable to open the message's CTL file.

zfErrE_FILE_CREATE_ERROR
Too many open files

zfErrE_FILE_ERROR
Could not read/write to file

zfErrE_FILE_NOT_FOUND
The file could not be found

zfErrE_FILE_OPEN_ERROR
Could not open file

zfErrE_FULLCOLLECTION
The collection is full

zfErrE_FUNCTION_ABORT
The function failed because a user defined callback function returned "Abort and Stop" status

zfErrE_INFO_FILE_ERROR
Error updating/accessing MSGDIR.CTL

zfErrE_INFO_FILE_INVALID
MSGDIR.CTL is invalid

zfErrE_INFO_FILE_OPEN_ERROR
Could not open MSGDIR.CTL

zfErrE_INIT_FAIL
The API failed to initialise

zfErrE_INVALID_DATA_FORMAT
The data file format specified is not recognised

zfErrE_INVALID_PARAM
An invalid parameter was passed to the Zetafax API

zfErrE_INVALID_ZF_INIT_FILE
The API had problems reading ZETAFAX.INI

zfErrE_INVALID_ZF_USER
An attempt was made to log on to the API with an invalid user

zfErrE_LICENCE_ERROR
The API Could not find the Zetafax licence

zfErrE_MALLOC_FAILED
The API ran out of memory.

zfErrE_MSG_EXISTS
This message already exists

zfErrE_MSG_NOT_COMPLETED
An attempt was made to modify a message that wasn't completed

zfErrE_MSG_UNKNOWN
The specified message could not be found

zfErrE_NOT_INIT
A call was made to the API when it had not been initialised with a call to ZfAPI.Logon

zfErrE_NO_START_SYSMAN
The API could not launch SYSMAN.EXE

zfErrE_PATH_NOT_FOUND
The API was passed an invalid path

zfErrE_REGISTRY_ERROR
Registry Error

zfErrE_SERVER_INI_FILE_ERROR
Problems accessing SETUP.INI

zfErrE_SERVER_NOT_RUNNING
The Zetafax server isn't running

zfErrE_SERVER_RUNNING
The Zetafax server is running

zfErrE_SUBMIT_FILE_INVALID
One or more lines in the specified .SUB file were invalid

zfErrE_TIMEOUT_EXPIRED
The specified function did not complete within the timeout period

zfErrE_TOO_MANY_FILES
There are too many files in this directory

zfErrE_USER_NOT_INIT
The API couldn't find the user of this session

Remarks

The error number may be returned as an IDispatch error. To see how these numbers map to the IDispatch errors and Zetafax's C API errors see the page Errors

Change History

This document details the change history of the COM API.

9.0.322.0

New Objects:

ZfLib.APIPrint - Enables API to print via Zetafax Printer

New Methods/Properties:

ZfLib.ZfAPI.SetZetafaxDirs - Enables API to override default logon server specified in the zfclient.ini file
ZfLib.ZfAPI.SetZetafaxServerFromAD - Enables API to override default logon server specified in the zfclient.ini file ZfLib.ZfAPI.GetZetafaxServersFromAD - Gets list of Zetafax servers registered in Active Directory
ZfLib.ZfAPI.GetZetafaxServerInfoFromAD - Gets server directory shares from Active Directory
ZfLib.ZfAPI.SystemDir - Returns System directory set by call to SetZetafaxDirs or SetZetafaxServerFromAD
ZfLib.ZfAPI.ServerDir - Returns Server directory set by call to SetZetafaxDirs or SetZetafaxServerFromAD

ZfLib.ZfAPI.UsersDir - Returns Users directory set by call to SetZetafaxDirs or SetZetafaxServerFromAD
ZfLib.ZfAPI.RequestDir - Returns Request directory set by call to SetZetafaxDirs or SetZetafaxServerFromAD
ZfLib.ZfAPI.ZetafaxServer - Returns Zetafax server set by call to SetZetafaxServerFromAD

Changes to Enumerations:

ZfLib .ZfErr - Added zfErrE_REGISTRY_ERROR, zfErrE_ACTIVE_DIRECTORY_NOT_PRESENT, zfErrE_ACTIVE_DIRECTORY_ERROR, zfErrE_ACTIVE_DIRECTORY_MISSING_VALUE

8.0.0.104

New Objects:

ZfLib.Coversheet - Holds the name of a coversheet
ZfLib.Coversheets - Contains a collection of Coversheet objects
ZfLib.Letterhead - Holds the name of a letterhead
ZfLib.Letterheads - Contains a collection of Letterhead objects

New Methods/Properties:

ZfLib.Message.MarkMsgAsRead - This method marks a message as read
ZfLib.MessageHistory.Name - Name of recipient associated with this MessageHistory item
ZfLib.MessageHistory.Organisation - Organisation of recipient associated with this MessageHistory item
ZfLib.MessageInfo.Organisation - Property containing the organisation of the first recipient of the message

ZfLib.MessageInfo.Type - Property containing the type of the Message (Fax, or SMS)
ZfLib.MessageInfo.UserStatus - Property containing the UserStatus of a message. Can be used to tell if the message has been read
ZfLib.NewMessage.Body - After the NewMessage.Send has been called this property contains the unique file body of the message
ZfLib.Recipients.AddSMSRecipient - This method adds an SMS recipient to the message
ZfLib.ServerInfo.Coversheets - Property that returns a collection of Coversheets

ZfLib.ServerInfo.Letterheads - Property that returns a collection of Letterheads ZfLib.UserSession.Logoff - Ends a session.
ZfLib.ZfAPI.LogonAnonymous - Creates an anonymous session

Changes to Enumerations:

ZfLib.UserStatusEnum - New enumeration used by MsgInfo.UserStatus
ZfLib.FaxTypeEnum - Added zfFaxTypeSMS
ZfLib .ZfErr - Added zfErrE_ACCESSDENIED and zfErrE_FULLCOLLECTION
ZfLib.StatusEnum - Added zfStatusPreviewOK and zfStatusPreviewFailed

Behavioural Changes:

COM+ - If available, the COM API now runs using the security call context

Permissions - The COM API now checks access permissions before attempting file operations. If the required permissions are not available the call will fail with zfErrE_ACCESSDENIED

There have also been changes to improve general efficiency and speed of some of the COM API's properties and methods

C language API

The C language Application Programmer's Interface (API) allows application programs to submit faxes to the Zetafax server for sending, and to control and monitor their progress as required.

A simple function call is provided to submit an ASCII text file in a given format. This format (termed Submit file format) includes details of the recipients of a message, in addition to the message text which can include text formatting such as bold and underlining. See The ZSUBMIT program for more on the Submit file format.

A range of other function calls allow other supported file formats to be submitted, and give full control over messages queued for a given user.

Libraries

The API comprises a set of C callable functions. It is distributed as a set of object libraries and a DLL (for use in different environments) for building into the application programs, together with a C header file containing definitions required to call the API functions. The following operating systems are currently supported:

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

Compilers

The object libraries for the API are available for use with the Microsoft C/C++ compiler.

File system structure

When the API is initialized you must specify a Zetafax username. Messages submitted by the API will appear as if submitted from a client logged on as this user, and the defaults for items such as the coversheet From name will be those for the user (exactly as if you submit a message from the client without changing any of the default values).

Store the message files to be sent in the user's out sub directory USERS*username*\\Z-OUT, where *username* is the username. If necessary API programs can use the ZfxGetUserOutDir function to get the name of this directory.

A message for sending consists of two files in the out directory: a control file and a data file. Both of these files, and all other files created by the server relating to this message, have the name *filename.ext* where *filename* identifies the message, and *ext* identifies the type of file. The *filename* is referred to in the API as the message body name, and this is supplied to the API functions when specifying messages to submit, delete, etc. The ZfxCreateAutoFile function is used to create a file with a unique body name in a given directory.

The control file is identified with a .CTL extension. This is an ASCII text file and gives full details of the options to be used in preparing a message for sending, including the coversheet, letterhead, and priority, together with details of the intended recipients of the message (name, organization, fax number, etc.). The file is updated by the server as the message is processed to give details of the progress for each addressee, errors, etc. For use with the API this file will generally have been created by interpreting a Submit format file.

The data file contains the message to be sent, and its extension depends on the format of the data.

Supported formats are as follows:

Extension	Format
.TXT	ASCII text
.EPN	Epson FX or Epson LQ Windows driver
.G3NTIFF	fax format normal resolution (200x100 dpi)
.G3FTIFF	fax format fine resolution (200x200 dpi)

Message information

When a message is submitted to the server for sending (after creating a control file and a data file for the message), an entry is made in the info file in the user's out directory. This entry includes the file name of the message, the comment associated with it, and the current status of the message (converting, sending, etc.). The server is then notified that a new message is ready for sending, and will add the message to its queue.

The status field is updated by the server program as the message is processed. When the server completes processing of the message and the status has changed to OK or FAILED the message may be deleted. This is done by removing the info file entry then deleting the message files, and may be done either from the client or via the API.

Details of a message in the info file may be obtained using the ZfxGetMsgInfo or ZfxGetMsgList routines, giving details of a single message or all messages in the info file respectively. These store the information in a data structure of type ZFMSGINFO. The fields in this structure are as follows:

Field	Structure
szBody	Character string giving the body name of the message files (i.e. the file name without extension or preceding period.). The maximum length of this string (excluding terminating NULL) is given by the constant ZFMSG_BODY_LEN.
szComment	Character string giving the description of the message being sent (as displayed on the right in the client main display). The maximum length of this string (excluding terminating NULL) is given by the constant ZFMSG_COMMENT_LEN.
Status	Current status of message. This is a variable of type ZFMSGSTATUS, and the value is one of the ZFMSG_??? values listed below.

The following list gives the possible states for a message and their meanings:

State	Meaning
ZFMSG_ADDING	Message being added to queue.
ZFMSG_HELD	Message held by user.
ZFMSG_DEFERRED	Message waiting for sending after a specific time, either because that time was specified when the message was submitted, or the user does not have sufficient permissions to send until after the given time.
ZFMSG_WAITING	Waiting for conversion or for a free device to send the message.
ZFMSG_CONVERTING	Preparing the message for sending (merging with letterhead, creating coversheet, e.t.c.).
ZFMSG_CONNECTING	Dialing remote device, or connecting to local printer, etc.

ZFMSG_SENDING	Connection made, sending message to remote device.
ZFMSG_SCANNING	Not applicable.
ZFMSG_ABORTING	Processing an Abort request, waiting for device controller or convert program to cancel processing of the message. ZFMSG_INCOMING Being received by the device.
ZFMSG_OK	Completed (sent) successfully, with no errors. The message may now be deleted (using the ZfxDeleteMsg function).
ZFMSG_FAILED.	Completed with one or more errors. Details of the errors which occurred are listed in the control file, although this would normally be checked manually. The message may now be deleted (using the ZfxDeleteMsg function)

Related topics

Function error returns and reference
Alphabetical reference
Message information functions
Message transmission history functions
Server and device status functions
Converting from older versions of the API

Converting programs from older versions of the Zetafax API

The API has been significantly extended over time. Existing API programs will continue to work correctly, and need not be rebuilt. However, any program which is to be rebuilt with the new version of the library will need some modification.

Version differences from versions 4.5 to 5 and 5 to 6 are detailed below.

For API 4.5 programs, a set of porting macros are included in the header file ZFAPI.H to allow programs to be ported with minimal changes (the addition of "#define" line before ZFAPI.H is included). They translate from the old function names to the new ones adding the necessary extra parameters. These are enabled by adding the following line before ZFAPI.H is included:

```
#define API_VER 0x450
```

For API 5 programs the superceded function names and structures are still supported by the DLLs and libraries, but by default are not included in the header file to assist new programmers. To continue using the old functions, add the following line before ZFAPI.H is included:

```
#define API_VER 0x500
```

However, the modifications required to reflect the changes described above are minor and quick to do. For clarity, it is therefore recommended that these changes are made in any programs which are still being developed or maintained, rather than relying on the porting macros.

Differences between version 4.5 and version 5

The API was significantly extended with Zetafax 5.

New features included:

- DLL version of library functions added
- Support for received faxes
- Retrieval of default user settings
- Support for multiple data files for each message
- Specifying the coversheet text in SUBMIT files

These enhancements made it necessary to make a number of changes to the original functions. The changes necessary for programs using version 4.5 are:

1. All functions have been renamed from Zf... to Zfx... (to ensure that any mismatching of old and new source code gives linker warnings).
2. Previously a function named ZfPrintError had to be written and was called directly by the API libraries. This function can now have any name, and is passed as a parameter to ZfxAPIInit. The prototype of the function has now changed to: void ZFAPICALLBACK Proc(CHAR FAR *)
3. All functions other than ZfxAPIInit and ZfxGetAPIVersion now take a session handle (of type ZFSESSIONHANDLE) as the first parameter. This is returned in the first parameter by the ZfxAPIInit function.
4. The following functions have an extra parameter of type ZFMSGDIR, to specify whether the function should act on the OUT or IN directory: ZfxAbortMsg, ZfxHoldMsg, ZfxReleaseMsg, ZfxDeleteMsg, ZfxCheckNewMsgStatus, ZfxGetMsgInfo, ZfxGetMsgList, ZfxGetMsgHistory
5. ZfxGetAPIVersion has an additional parameter to return an integer value giving the version number. This can be NULL if not required.
6. The functions ZfSendSubmitFile, ZfCreateCtlFile and ZfCreateDataFile have been changed so they now take file names (CHAR FAR *) instead of file pointers (FILE *) to allow them to be used in the DLL version of the library. Three functions ZfxSendSubmitFileFP etc have been supplied in the static library version of the API to simplify porting of existing applications; these take file pointers as before.

Differences between version 5 and version 6

The API has been extended again for Zetafax 6.

New features include:

- Support for the subject line in faxes
- Inclusion of Least Cost Routing information in message status, message history, and server status calls
- Info structures and functions extended

These enhancements made it necessary to add new some new functions to the API. The new functions are similar to the original functions, but include extra parameters for newly supported features.

The following structures are new, replacing the equivalent structures without the "EX" ending:

ZFMSGINFOEX
ZFMSGHISTORYEX
ZFSERVERINFOEX

The following functions are new, replacing the equivalent functions without the "Ex" ending:
ZfxGetServerStatusEx
ZfxGetMsgHistoryEx
ZfxGetMsgInfoEx
ZfxGetMsgListEx

Related topics
Function error returns and reference
Alphabetical reference
Message information functions
Message transmission history functions
Server and device status functions

Function overview

This section gives a brief overview of the functions which comprise the API.

User supplied functions

If an API function encounters an error, it can call a user-supplied error routine with a textual error string before returning an error code to the calling program. This can be particularly useful when writing or testing the application.

General routines

Before using any API functions the program should call ZfxAPIInit. This specifies the username to use and returns a handle which is used in subsequent calls. It may be called more than once with different usernames if required. The ZfxAPIClosedown routine should be called by the program for each handle returned by ZfxAPIInit before exiting. The ZfxGetAPIVersion function returns the version number of the library (for printing in a startup message). ZfxCreateAutoFile generates a file in a given directory so that no other files exist with the same body name. Finally ZfxCheckServer checks whether the Zetafax server is running (although several of the other routines will also return an error if the Zetafax server is not running when they are called).

User information

Messages submitted by API programs use the Zetafax user's default settings for any message options which are not specified directly. The ZfxGetUserCoversheet and ZfxGetUserFromname functions return the user's default coversheet and sender name, and are used when a program wants to change its behavior depending on these settings.

Location of files

The standard Zetafax installation gives each user a base sub directory called USERS*username* where *username* is the username. Message files for sending and received messages are stored by the server in sub directories USERS*username*\\Z-OUT and USERS*username*\\Z-IN respectively. The full pathnames of these directories are returned by the ZfxGetUserArea, ZfxGetUserOutDir and ZfxGetUserInDir functions respectively. Although not normally needed, the ZfxGetSystemArea routine may be used to obtain the full pathname of the SYSTEM sub directory, used for storing letterheads, etc.

Submitting a message

The simplest method of submitting a fax for sending is to create an ASCII text Submit file containing both the message options and the message text, then call the ZfxSendSubmitFile function. This creates a control and data file in the user's out directory, then submits the message to the server (which also makes an entry in the info file for that user).

Alternatively these three stages may be carried out independently. The ZfxCreateCtlFile function interprets the %%[MESSAGE] section of a Submit file, which contains details of the message options and addresses, and generates a control file. The ZfxCreateDataFile interprets the %%[TEXT] section of a Submit file to generate an ASCII text or Epson FX format data file if required (or a file of the correct format could simply be copied or created by the program). Finally the ZfxSendMsg function makes an entry in the info file and submits a control and data file pair to the server.

Programs which are unable to store a session handle for passing to the other routines can use ZfxVBSendSubmitFile. This is a self contained function which initializes an API session, submits a file, then closes the session again, and is ideal for calling from Visual Basic or macro languages.

There are three additional routines ZfxSendSubmitFileFP, ZfxCreateCtlFileFP and ZfxCreateDataFileFP which are provided to assist in porting programs written for older versions of the API. Their functions are the same as their namesakes above, but they use C file pointers instead of file names and are therefore

only supported in the static libraries, not in the DLL.

Server status

ZfxCheckServer checks whether the fax server is running (although several of the other routines will also return an error if the fax server is not running when they are called). ZfxGetServerStatusEx returns information about the server, including the number of queued messages in various states, the status of each of the devices, and the status of any links to remote servers.

Server control

ZfxStartServer, ZfxStopServer and ZfxRestartServer can be used to control the Zetafax server programs for completely automated systems.

Message control

Once a message has been submitted to the Zetafax server, its status may be obtained using the ZfxGetMsgInfo function. When the message completes, this function also specifies whether it was successfully transmitted. The ZfxGetMsgHistory function can be used to retrieve the full transmission history for the message, including details such as the number of dial attempts made and the connection time.

The ZfxGetMsgList function returns the status of all messages currently queued (or completed but not yet deleted) for the given user. The ZfxCheckNewMsgStatus function is used in conjunction with this - it checks whether the status of any messages for the user have changed since it was last called to save the overhead involved in calling ZfxGetMsgList unnecessarily.

The ZfxAbortMsg function requests the Zetafax server to abort processing of a given message (i.e. stop preparation or transmission of the message). When the server has completed processing it sets the message status to OK or FAILED. Messages in either of these states may be deleted from the info file by calling ZfxDeleteMsg. Optionally, this will also delete the control and data files for the message, together with any other temporary files generated by the server. The ZfxDeleteMsg function can also be used to delete all messages for a given user.

The ZfxHoldMsg and ZfxReleaseMsg functions can be used to pause and then resume processing for a specific message. Held messages are not submitted to devices for sending, though they retain their position in the queue.

These functions, except ZfxAbortMsg, ZfxHoldMsg, ZfxRushMsg and ZfxReleaseMsg, can also be used for received messages. API licensing

The API must be licensed on each Zetafax system on which you wish to run programs which use it (including the ZSUBMIT program). Call Equisys or your supplier for price details of multiple licenses.

Related topics

Function error returns and reference
Alphabetical reference
Message information functions
Message transmission history functions
Server and device status functions
Converting from older versions of the API

Message information

When a message is submitted to the server for sending (after creating a CONTROL file and DATA file for the message), an entry is made in the INFO file in the user's OUT directory. This entry includes the file name of the message, the comment and subject line associated with it, and the current status of the message (converting, sending etc). The server is then notified that a new message is ready for sending, and will add the message to its queue.

The status field is updated by the server program as the message is processed. When the server completes processing of the message (and the status has changed to OK or FAILED) the message may be deleted. This is done by removing the INFO file entry, then deleting the message files, and may be done either from the client or via the API.
Details of a message in the INFO file may be obtained using the ZfxGetMsgInfoEx or ZfxGetMsgListEx routines, giving details of a single message or all messages in the INFO file respectively. These store the information in a data structure of type ZFMSGINFOEX. The fields in this structure are as follows.

Field	Description
Status	Current status of message. This is a variable of type ZFMSGSTATUS, and the value is one of the ZFMSG_??? values listed below. Message status variables are given below.
szBody	Character string giving the body name of the message files (ie the file name without extension or preceding '.'). The maximum length of this string (excluding terminating NULL) is given by the constant ZFMSG_BODY_LEN.
szComment	Character string giving the description of the message being sent (as displayed on the right in the client main display). The maximum length of this string (excluding terminating NULL) is given by the constant ZFMSG_COMMENT_LEN.
szSubject	Character string giving the subject of the message being sent (as displayed on the right in the client main display). The maximum length of this string (excluding terminating NULL) is given by the constant ZFMSG_SUBJECT_LEN.
UserStatus	Current status of message in relation to the user. This is a variable of type ZFMSGUSERSTATUS, and the value is one of the ZFMSG_??? values listed below. User status variables are given below.
Type	Type of message. This is a variable of type ZFMSGTYPE, and the value is one of the ZFTYPE_??? values listed below. Type variables are given below.
szOrganisation	Character string giving the

	organisation field for the message being sent. The maximum length of this string (excluding terminating NULL) is given by the constant ZFMSG_COMMENT_LEN.

Message status

The following list gives the possible states for a message (variable of type ZFMSGSTATUS).

Message state	Description
ZFMSG_ADDING	Message being added to queue
ZFMSG_HELD	Message held by user
ZFMSG_DEFERRED	Message waiting for sending after a specific time, either because that time was specified when the message was submitted, or the user does not have sufficient permissions to send until after the given time.
ZFMSG_WAITING	Waiting for conversion or for a free device to send the message.
ZFMSG_CONVERTING	Preparing the message for sending (merging with letterhead, creating coversheet etc).
ZFMSG_CONNECTING	Dialling remote device, or connecting to local printer etc.
ZFMSG_SENDING	Connection made, sending message to remote device.
ZFMSG_SUB_ROUTER	Passed to a remote Zetafax server for sending, but not yet acknowledged.
ZFMSG_ACCEPT_REMOTE	Message accepted for transmission by a remote Zetafax server.
ZFMSG_SCANNING	(not applicable)
ZFMSG_ABORTING	Processing an Abort request, waiting for device controller or convert program to cancel processing of the message.
ZFMSG_INCOMING	Being received by the device.
ZFMSG_WAITING_SCAN_DOC	(not applicable)
ZFMSG_OK	Completed (sent or received) successfully, with no errors. For sent messages the message may now be deleted (using the ZfxDeleteMsg function). For received messages the DATA file contains the received message, so should be saved (if required) before calling ZfxDeleteMsg.
ZFMSG_FAILED	Completed with one or more errors. Details of the errors that occurred are listed in the CONTROL file, although this would normally be checked manually. The message may now be deleted (using the ZfxDeleteMsg function). For received faxes any part of the message received before the error occurred will still be stored in the DATA file.
ZFMSG PREVIEW OK	(not applicable)
ZFMSG PREVIEW FAILED	(not applicable)

User status

The following list gives the possible users states for a message (variable of type ZFMSGSTATUS).

Message state	Description
ZFMSG_U_UNKNOWN	User Status unknown.
ZFMSG_U_OK	Message has been read by user.
ZFMSG_U_AWAITING	Message awaiting preview.
ZFMSG_U_WAITING	Waiting - i.e. unread.

User status

The following list gives the possible values for the message type field (variable of type ZFMSGTYPE).

Type	Description
ZFTYPE_FAX	Fax message.
ZFTYPE_LAN	Message for another Zetafax user (on the same LAN).
ZFTYPE_SMS	Text message.

top

Related topics

Function overview
Function error returns and reference
Alphabetical reference
Message transmission history functions
Server and device status functions
Converting from older versions of the API

Message transmission history

Details of the transmission history for a message in the CONTROL file may be obtained using the ZfxGetMsgHistoryEx routine, giving details of a single addressee or all addressees. This stores the information in a data structure of type ZFMSGHISTORYEX. The fields in this structure are as follows:

Below you will find details for:
Message transmission history Message event types Message routes

Message transmission history

Field	Description
EventType	Type of this record. This is a variable of type ZFMSGEVENT, and the value is one of the ZFEVENT_??? values listed below.
Year	Event timestamp (range 1980 to 2079)
Month	Event timestamp (range 1 to 12)
Day	Event timestamp (range 1 to 31)
Hours	Event timestamp (range 0 to 23)
Mins	Event timestamp (range 0 to 59)
Secs	Event timestamp (range 0 to 59)
AddrNum	Addressee number (starts from 1). When a single message has been addressed to more than one person, this allows the events to be identified.
Route	Route type used. This is a variable of type ZFMSGROUTE, and the value is one of the ZFROUTE_??? values listed below.
szRouteParams	Route parameters used. For fax routes this is the fax number dialled. The maximum length of this string (excluding terminating NULL) is given by the constant ZFDEV_PARAMS_LEN.
ErrorCode	Reason for failure. This is a variable of type ZFERR. The error code can be any of the L2ERR_??? values given in the file ZFERR.H - however note that the values depend on the version of the Zetafax server running, not the version of the API used. New versions of the server will have additional error codes.
PagesSent	Number of last page successfully sent (including the coversheet if used). If a three page message had several attempts at sending, but only the first two pages were sent successfully, this value would be set to 2. For event type ZFEVENT_SENT_OK this is the total number of pages in the message.
ConnectSecs	Connection time in seconds. This field only applies to events of type ZFEVENT_SENT_OK and ZFEVENT_SENT_ERROR, and gives the total connect time for the given addressee.
szDevice	The name of the device used to send the message. The maximum length of this string (excluding terminating NULL) is

	given by the constant ZFDEV_NAME_LEN.
szRemoteServer	The name of the remote server used to send the message via LCR. The maximum length of this string (excluding terminating NULL) is given by the constant ZFDEV_NAME_LEN.

top

Message event types

The following list gives the possible event types for a message (variable of type ZFMSGEVENT).

Event	Description
ZFEVENT_SENT_OK	Sent successfully
ZFEVENT_RECD_OK	Received successfully
ZFEVENT_SCAN_OK	Scanned successfully
ZFEVENT_SENT_ERROR	Sent with errors - the connection time and number of pages can be used to determine whether the error was during connection or transmission
ZFEVENT_RECD_ERROR	Received with errors.
ZFEVENT_SCAN_ERROR	Scanning request completed with errors.
ZFEVENT_TRIED	Send attempt unsuccessful. The Zetafax server makes several attempts to send to a given addressee. This event is logged if a send attempt is unsuccessful, but the server will retry later. If all attempts are unsuccessful, the last attempt will be logged as an event of type ZFEVENT_SENT_ERROR, while all previous events are logged as ZFEVENT_TRIED events.
ZFEVENT_ROUTER_SUB	Passed to remote Zetafax server for sending (when Least Cost Routing in use)
ZFEVENT_ROUTER_ACC	Accepted for transmission by remote server
ZFEVENT_ROUTER_ERR	Submission to a remote server failed

top

Message routes

The following list gives the possible routes for fax messages (variable of type ZFMSGROUTE). Note that other routes are also supported by the Zetafax server, so this list is not exhaustive.

Route	Description
ZFROUTE_FAX_NORMAL	Standard resolution fax (200 x 100 dpi)
ZFROUTE_FAX_FINE	Fine resolution fax (200 x 200 dpi)

top

Related topics

Function overview
Function error returns and reference

Message defaults

This structure conatins the default message settings for the user. It is used in the `ZfxGetMsgDefaultsEx()` function

Field	Description
PriorityDefault priority.	This is a variable of type ZFMSGPRIORITY, and the value is one of the ZFPRIORITY_??? values listed below. Priority variables are given below.
HeaderDefault header.	This is a variable of type ZFMSGHEADER, and can be a combination of one or more of the ZFHEADER_??? values listed below. Header variables are given below. These values should be combined using the bitwise-OR operator
QualityDefault Quality.	This is a variable of type ZFMSGQUALITY, and the value is one of the ZFQUALITY__??? values listed below. Quality variables are given below.
SendTime	Variable of type ZFSENDTIME determining when a message submitted by the user will be sent. The value is one of the ZFSENDTIME__??? values listed below. Send Time variables are given below.
AfterYear, AfterMonth, AfterDay, AfterHour, AfterMin, AfterSec;	Short integer values specifying the 'send after' time. These parameters are ignored unless the SendTime parameter is set to ZFSENDTIME_AFTER.
szFrom	Character string giving the from field to use. The maximum length of this string (excluding terminating NULL) is given by the constant ZFMSG_FULLNAME_LEN.
szCoversheet	Character string giving the name of the coversheet to use. The maximum length of this string (excluding terminating NULL) is given by the constant ZFMSG_COVERSHEET_LEN.
szLetterhead	Character string giving the name of the letterhead to use. The maximum length of this string (excluding terminating NULL) is given by the constant ZFMSG_LETTERHEAD_LEN.

Priority

The following list gives the possible values for message priority.

Message priority	Description
ZFPRIORITY_BACKGROUND	The lowest priority - messages with this priority will be processed after messages with other priorities.
ZFPRIORITY_NORMAL	Standard priority
ZFPRIORITY_URGENT	The highest priority - messages with this priority will take precedance over messages with other priorities.

Message header

The following list gives the possible message header settings (variable of type ZFMSGHEADER).

Message header setting	Description
ZFHEADER_NONE	No setting.
ZFHEADER_NUMBER	Phone number.
ZFHEADER_TO	To field.
ZFHEADER_FROM	From field.
ZFHEADER_DATE	Date field.
ZFHEADER_TIME	Time field.

Quality

The following list gives the possible values for message quality.

Message quality	Description
ZFQUALITY_DRAFT	Low quality.
ZFQUALITY_NORMAL	Standard quality.
ZFQUALITY_HIGH	High quality.

Send time

The following list gives the possible values for the SendTime field.

Send time	Description
ZFSENDTIME_NOW	Send the message immediately.
ZFSENDTIME_OFFPEAK	Send the message during the offpeak period.
ZFSENDTIME_AFTER	Send the message after the time specified by the 'send after' fields.

top

Related topics

Function overview
Function error returns and reference
Alphabetical reference
Message transmission history functions
Server and device status functions
Converting from older versions of the API

Server and device status

Details of the status of the server and the configured devices and links may be obtained using the ZfxGetServerStatusEx function. It is passed a structure of type ZFSERVERSTATUSEX which defines what information is required. The structure includes the addresses of three further structures of type ZFSERVERINFOEX, ZFDEVICEINFOEX and ZFLINKINFOEX, which are used to return the information.

Below you will find information on:

Server information Server status
Device information Device status
Link information Link status

Server status

Field	Description
ServerInfoExSize	Should be set to sizeof (ZFSERVERINFOEX)
lpServerInfoEx	Address of a SERVERINFOEX structure, to be filled on return
MaxDevices	Maximum number of device entries to be returned (i.e. number of elements in the DeviceInfo array)
lpNumDevices	Address of short integer variable used to return the number of devices in the file. **Note:** that the returned value may be larger than the value given for MaxDevices if the buffer was not large enough for all entries.
DeviceInfoEx	Size Should be set to sizeof (ZFDEVICEINFOEX)
IpDeviceInfoEx	Address of array of 'n' ZFDEVICEINFOEX structures, where 'n' is the value given by the MaxDevices parameter.
MaxLinks	Maximum number of LCR link entries to be returned (ie number of elements in the LinkInfoEx array)
lpNumLinks	Address of short integer variable used to return the number of links in the file. Note that the returned value may be larger than the value given for MaxLinks if the buffer was not large enough for all entries.
LinkInfoExSize	Should be set to sizeof (ZFLINKINFOEX)
lpLinkInfoEx	Address of array of 'n' ZFLINKINFOEX structures, where 'n' is the value given by the MaxLinks parameter.

top

Server information

The fields in the ZFSERVERINFOEX structure are as follows:

Field	Description
Queue	Deferred Number of items in queue waiting until a given time (excluding those waiting before retrying after failure).
QueueWaitingResend	Number of items in queue waiting before retrying after a send attempt has failed.
QueueWaitingConvert	Number of items in queue waiting to be converted (prepared for sending).
QueueConverting	Number of items in queue being converted (prepared for sending) - currently either 0 or 1.
QueueWaitingDevice	Number of items in queue waiting for a device to become available
QueueSending	Number of send requests in queue being processed by a device - either connecting or sending.
QueueScanning	Number of scan requests items in queue being processed by a scanning device.
QueueSubRouter	Number of items submitted to Router for sending, and awaiting acceptance by a remote server.
QueueAcceptRemote	Number of items currently accepted for sending by remote servers

top

Device information

The fields in the ZFDEVICEINFOEX structure are as follows:

Field	Description
szDevice	Name of the device. Device names comprise a device type and device number, separated by a hyphen (eg "FCLASS-3"). Two devices of different types can have the same number. The maximum length of this string (excluding terminating NULL) is given by the constant ZFDEV_NAME_LEN.
Status	Current status of the device. This is a variable of ZFDEVSTATUS, and the value is one of the ZFDEV_??? values listed below.
szUser	User name of owner of message currently being processed by the device (or

	null string if no message being processed). The maximum length of this string (excluding terminating NULL) is given by the constant ZFUSER_NAME_LEN.
szMsgBody	Body name of message CONTROL file currently being processed by the device (or null string if no message being processed).
NumPages	Number of pages in message currently being processed (including coversheet if used)
CurrentPage	Page number currently being sent.
NumConnect Fails	Number of send attempts by this device which have failed to connect.
NumSendFails	Number of send attempts by this device which have failed after connection.
NumSentOK	Number of messages sent successfully by the device.

top

Device status

The following list gives the possible status values for a device (variable of type ZFDEVSTATUS).

Field	Description
ZFDEV_UNKNOWN_STATE	Status unknown (eg not initialized yet). This could also occur if an API program is run with a later version of the Zetafax server which supports additional device states.
ZFDEV_IDLE	Ready for sending
ZFDEV_ERROR	Trying to recover from error
ZFDEV_FAILED	Unable to recover from error. The device status changes to ZFDEV_FAILED when the device is unable to recover from a status of ZFDEV_ERROR (generally after a fixed period of time).
ZFDEV_OFFLINE	Device set offline by user, and unavailable for sending
ZFDEV_BUSY	Generally busy, and unavailable for sending
ZFDEV_WAITING_CONNECT	Waiting for connection (for fax messages, this is the status between dialing the fax number and completing the fax handshake). Note that for dialed calls, the connection time starts from when the modem reports that the remote device has answered the call - the device status will generally be ZFDEV_WAITING_CONNECT for several seconds after this.
ZFDEV_SENDING	Connected to the remote device and sending the message.
ZFDEV_RECEIVING	Receiving an incoming message
ZFDEV_POLL_SEND	Not supported
ZFDEV_POLL_RECEIVE	Not supported
ZFDEV_WAITING_SCAN_DOC	For scanning requests, waiting for

	the document to be scanned to be placed on the device.
ZFDEV_SCANNING	Not supported
ZFDEV_INCOMING_CALL	Answering an incoming call - state will change to ZFDEV_RECEIVING once the correct handshaking has occurred

top

Link information

The fields in the ZFLINKINFOEX structure are as follows:

Field	Description
szRemoteServer	Name of the remote server to which the link is configured. The maximum length of this string (excluding terminating NULL) is given by the constant ZFDEV_NAME_LEN.
RemoteLinkStatus	Current status of the remote server on the link. This is a variable of type ZFLINKSTATUS, and the value is one of the ZFLINK_??? values listed below.
LocalLinkStatus	Current status of the local server on the link. This is a variable of type ZFLINKSTATUS, and the value is one of the ZFLINK_??? values listed below.
fConnectionOK	Current status of the connection (mail or WAN) to the remote server. This is a Boolean value, 1 for a good connection, 0 for no connection.
fLinkActive	This is a Boolean value indicating whether the link is active, and available for sending and receiving (1 if the link is active, 0 if the link is not active). This value is computed from RemoteLinkStatus, LocalLinkStatus and fConnectionOK fields.
NumSentOK	Number of messages sent successfully by the remote server.
NumTimedOut	Number of messages timed out waiting for responses over the link.
NumRejected	Number of messages rejected by the remote server.
NumDeviceError	Number of messages failed to be sent by the remote server as a result of device errors.
NumRemoteServerErr	Number of messages failed to be sent by the remote server as a result of other errors at the remote server.
NumUnack	Number of messages submitted to the remote server and still awaiting acknowledgement.
NumAck	Number of messages submitted to the remote server and acknowledged.
NumReceived	Number of messages received from

	the remote server for sending locally.

top

Link status

The following list gives the possible status values for a link (variable of type ZFLINKSTATUS).

Field	Description
ZFLINK_UNKNOWN_STATE	Remote link status unknown. This occurs when the Zetafax server has not received any communication from the remote server.
ZFLINK_INITIALISING	Link is currently being initialised.
ZFLINK_ONLINE	Link is online.
ZFLINK_OFFLINE	Link has been set to offline by the administrator, and is unavailable for sending.
ZFLINK_FAILED_WDOG	The link watchdog failed to receive an expected message.
ZFLINK_SHUTDOWN	The link has been shut down.
ZFLINK_UNRECOGNISED_STATE	Unrecognised state. This could occur if an API program is run with a later version of the Zetafax server which supports additional link states.

top

Related topics

Function overview
Function error returns and reference
Alphabetical reference
Message information functions
Message transmission history functions
Converting from older versions of the API

Function error returns and reference

Return	Description
ZFERR_BUFFER_TOO_SMALL	One of the buffer lengths specified in the call is smaller than the length of the string which is to be returned. Increase the buffer length.
ZFERR_CANNOT_LOG_ON	The user specified in the ZfxAPIInit call is already logged on (and the 'exclusive' option was specified), or the program has insufficient access permissions to the user directories.
ZFERR_CANNOT_READ_MSG_ DEFAULTS	Unable to read the default settings for this user contained in the USER. INI files. Check the program has read access to the SYSTEM\Z-DB directory and the user's Z-DB directory, and that these files exist.
ZFERR_CANNOT_RUN_SYSTEM_ MANAGER	Error running the system manager program SYSMAN.EXE. Check the file exists in the ServerArea (as specified in the ZETAFAX.INI file)
ZFERR_CANNOT_SUBMIT_ REQUEST	Error submitting request to the Zetafax server. Check access permissions to the REQUEST directory.
ZFERR_CONTROL_FILE_ERROR	Error reading or writing CONTROL file.
ZFERR_CONTROL_FILE_INVALID	CONTROL file is corrupt.
ZFERR_CONTROL_FILE_OPEN_ERROR	Unable to open the CONTROL file. Check the file exists, and that the program has sufficient access permissions.
ZFERR_ERROR_STARTING_SERVER	General error starting the server. The error is logged to the SERVER. LOG file in the SERVER\Z-DB directory.
ZFERR_FILE_CREATE_ERROR	Unable to create the specified file. Check the path and file name are valid, and that the program has sufficient access rights to the directory.
ZFERR_FILE_ERROR	Unable to read from or write to the specified file. Check the path and file name are valid, the file exists, and that the program has sufficient access rights to the directory.
ZFERR_FILE_NOT_FOUND	The specified file does not exist. Check the path and file name are valid, the file exists, and that the program has sufficient access rights to the directory.
ZFERR_FILE_OPEN_ERROR	Unable to open the specified file. Check the path and file name are valid, the file exists, and that the program has sufficient access rights to the directory.
ZFERR_FUNCTION_ABORTED	The function has failed because a user supplied callback function has

	returned an "abort and stop waiting" status.
ZFERR_INFO_FILE_ERROR	Error reading or writing MSGDIR.CTL file.
ZFERR_INFO_FILE_INVALID	MSGDIR.CTL file is corrupt.
ZFERR_INFO_FILE_OPEN_ERROR	Unable to open the MSGDIR.CTL file. Check the file exists, and that the program has sufficient access permissions.
ZFERR_INVALID_DATA_FORMAT	The data file format specified is unknown. Check the parameter and file type.
ZFERR_INVALID_PARAMETERS	One or more of the parameters given to the routine has an invalid format (eg file name too long). Check the parameters.
ZFERR_INVALID_ZF_INIT_FILE	The server directories could not be determined from the ZETAFAX.INI file. Check the file has not been corrupted, and if necessary reinstall the server or client on the PC.
ZFERR_MESSAGE_ALREADY_EXISTS	The message specified in the ZfxSendMsgEx call is already referenced in the INFO file. This can happen if a message file is deleted manually without calling ZfxDeleteMsg to delete the references, or if an existing control file is overwritten by a new one created without using the ZfxCreateAutoFile routine to ensure no files of the given name already exist. Call ZfxAbortMsg if the status is not OK or FAILED (to delete the message from server queues), call ZfxDeleteMsg and try again.
ZFERR_MESSAGE_NOT_COMPLETED	The message specified in the ZfxDeleteMsg call is still being processed by the server. Call ZfxAbortMsg first, then wait for the state to change to ZFMSG_OK or ZFMSG_FAILED.
ZFERR_NO_ZF_INIT_FILE	Initialization file ZETAFAX.INI not found. The directory containing this is normally given in the environment variable ZFAXINIT. Check that this variable exists in the environment - for an OS/2 program being run detached or from the program manager the SET ZFAXINIT= line must have been added to the CONFIG.SYS file, and the PC rebooted before the change will be noted.
ZFERR_NOT_INITIALISED	The ZfxAPIInit routine has not been called successfully. This must be done before calling the given routine.
ZFERR_PATH_NOT_FOUND	The given path or file does not exist ZFERR_SERVER_INI_FILE_ERROR Server initialisation file SETUP.INI not found or invalid.
ZFERR_SERVER_NOT_RUNNING	The Zetafax server is not running, or the program is unable to communicate with the server. Check that the server has not been stopped, and that the program has

	sufficient access to the server REQUEST directory.
ZFERR_SERVER_RUNNING	Some or all of the server programs are already running. The server must be completely stopped before the ZfxStartServer function or ZfxDeleteMsg function (specifying all messages) can be used.
ZFERR_SUBMIT_FILE_INVALID	One or more lines in the SUBMIT file specified are invalid. The error text given in the ZfxPrintError call details the line at fault.
ZFERR_TIMEOUT_EXPIRED	The specified function has not completed within the timeout period.
ZFERR_TOO_MANY_FILES	Too many files (ie 1000) of the format specified in the ZfxCreateAutoFile call already exist, and a new one can not be created.
ZFERR_UNKNOWN_MESSAGE	The control file for the message specified does not exist, or it is not referenced in the INFO file.
ZFERR_UNKNOWN_USER	The Zetafax username given in the ZfxAPIInit call has not been configured on the Zetafax system.

Related topics

Function overview
Alphabetical reference
Message information functions
Message transmission history functions
Server and device status functions
Converting from older versions of the API

Alphabetical reference

There follows a description of all of the routines available within the Zetafax C language API. Each routine includes some example code showing how it may be called from within your application.

ZfxGetUserInDir

ZfxGetUserOutDir

ZfxHoldMsg

ZfxMarkMsgAsRead

ZfxReleaseMsg

ZfxRestartServer

ZfxRushMsg

ZfxSendMsg

ZfxSendMsgEx

ZfxSendSubmitFile

ZfxSendSubmitFileFP

ZfxStartServer

ZfxStopServer

ZfxVBSendSubmitFile

Related topics
Function error returns and reference
Message information functions
Message transmission history functions
Server and device status functions
Converting from older versions of the API

User supplied error message display function

Syntax

```
void ZFAPICALLBACK MyError( char FAR *lpszErrorText)
```

Parameters

Parameter	Description
lpszErrorText	Null terminated string containing explanatory text for error

Description

This routine should be defined within the application program itself - it can have any name. Its address is passed as one of the parameters to ZfxAPIInit , and it is called by the API before the API routines return certain error codes, to give additional text explaining why the call failed. For example, if a SUBMIT file is rejected the API will call this routine with the reason, before returning the error code ZFERR_SUBMIT_FILE_INVALID.

For programs which interact with the user the text is probably best displayed on screen, as an additional help to any error message which the program itself may display as a result of the returned error code. Other programs may wish to log the error to disk or ignore it (defining an empty function) as required. Note that a FAR (32-bit) pointer to the string is supplied - in 16 bit environments the "%Fs" (or equivalent) format specifier will be needed in calls to printf etc.

Return Value

There is no return value.

Example

```
#include <stdio.h>
#include <zfapi.h>
...
void ZFAPICALLBACK DisplayAPIError( char FAR *lpszText)
{
      /* Note - remember to use %Fs in 16 bit */
      /* programs because it is a FAR pointer */
      printf("*API ERROR* %Fs\n", lpszText);
}
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxAbortMsg

Gets the default message settings for the user.

Syntax

```
ZFERR FAR ZfxAbortMsg( ZFSESSIONHANDLE hSession, ZFMSGDIR MsgDir, char FAR  *lpszBody)
```

Parameters

Parameter	Description
Hsession	API session handle, as returned by ZfxAPIInit call
MsgDir	Message type (ZFDIR_OUT)
lpszBodyBase	name of message file

Description

This routine is called to stop the server processing a message after it has been submitted using ZfxSendMsgEx . The routine sends a request to the Zetafax server, which processes it asynchronously. The routine can be called even if the server has already finished processing the message.

Return value

The routine returns 0 if successful, otherwise one of the following:

ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_UNKNOWN_MESSAGE
ZFERR_SERVER_NOT_RUNNING
ZFERR_CANNOT_SUBMIT_REQUEST

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
 if (ZfxAbortMsg(hSession, ZFDIR_OUT, "~XSND000") == 0)
 {
 printf("Message aborted\n");
        /* now wait for status to change to */
        /* ZFMSG_OK or ZFMSG_FAILED before */
       /* deleting the message */
 ...
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxAPIClosedown

Closedown API routines.

Syntax

```
ZFERR FAR ZfxAPIClosedown( ZFSESSIONHANDLE hSession)
```

Parameters

Parameter	Description
hSessionAPI	session handle, as returned by ZfxAPIInit call

Description

This routine should be called for each session handle returned by ZfxAPIInit . It releases any resources used by the API routines.

Return value

The routine returns 0.

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
 ZFSESSIONHANDLE hSession;
if (ZfxAPIInit(&hSession, "FRED", TRUE,  MyErrorProc) == 0)
       {
       /* call other API functions */
       ...
       /* now cleanup */
       ZfxAPIClosedown(hSession);
       }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxAPIInit

Initialise Zetafax API.

Syntax

```
ZFERR FAR ZfxAPIInit( ZFSESSIONHANDLE *phSession, char FAR *lpszUserName, short
fExclusive, ZFERRORPROC *lpErrorProc)
```

Parameters

Parameter	Description
phSession	Address of variable of type ZFSESSIONHANDLE, used to return the API session handle. This handle should then be passed to other API functions to identify the session.
lpszUserName	Zetafax username to use when submitting messages fExclusiveZero if this user is permitted to be logged on elsewhere (either on a normal Zetafax client, or in another API program).
lpErrorProc	Address of error message callback function, called when an error occurs with a text string if there is additional information relating to the error. This can be set to NULL if not required.

Description

This routine should be called before calling any of the other API functions. The routine may be called more than once by the program if it wishes to log on as more than one user. The paired routine ZfxAPIClosedownshould be called for each handle returned by this function before exiting from the program.
If the fExclusive flag is set (non-zero), then each call to this function may result in a "lock" file being kept open until ZfxAPIClosedown is called. Because most programs have a limit to the total number of files which may be kept open, it is recommended that you only have one session open at a time if fExclusive is set - ie ZfxAPIClosedown is always called before calling ZfxAPIInit again.

Return value

The routine returns 0 if successful, otherwise one of the following:

```
ZFERR_INVALID_PARAMETERS
ZFERR_NO_ZF_INIT_FILE
ZFERR_INVALID_ZF_INIT_FILE
ZFERR_UNKNOWN_USER
ZFERR_CANNOT_LOG_ON
```

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...

ZFERR Err;
ZFSESSIONHANDLE hSession;

/* Log on as FRED, which sets hSession if ok */

Err = ZfxAPIInit(&hSession, "FRED", TRUE,  MyErrorProc);
if (Err == 0)
 {
       /* call required functions */
      ...
      /* cleanup */
      ZfxAPIClosedown(hSession);
 }
 else
 {
      printf("Unable to log in as FRED\n");
 }
```

Related topics

Alphabetical reference
Function error returns and reference

ZfxCheckNewMsgStatus

Check if message status has changed.

Syntax

ZFERR FAR ZfxCheckNewMsgStatus(ZFSESSIONHANDLE hSession, ZFMSGDIR MsgDir, short FAR *lpfStatusChanged)

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call
MsgDir	Message type - ZFDIR_OUT for sent messages, or ZFDIR_IN for received messages
lpfStatus Changed	Address of short integer boolean variable which is set to a non-zero value if the status of one or more messages may have changed since the last call to this function, 0 otherwise 1.

Description

This routine checks whether the status of any of the messages in the current users OUT or IN directory (depending on the setting of MsgDir) has changed. If the fStatusChanged variable is set (non-zero) on return then the calling program should call the ZfxGetMsgInfoEx routine for each message it is interested in (or the ZfxGetMsgListExto get information for all messages) to identify what (if anything) has changed.

This function is supplied because ZfxGetMsgInfoEx and ZfxGetMsgListEx stop the server updating the status of any messages for that user while running, and should therefore only be called when necessary. The ZfxCheckNewMsgStatus function acts by checking the attributes of a file, and therefore it is recommended that it is not called more frequently than every 5 to 10 seconds unless a quicker response is required.

Return value

The routine returns 0 if successful, otherwise one of the following:

ZFERR_NOT_INITIALISED
ZFERR_INFO_FILE_ERROR

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
 for (;;)
 {
       if (ZfxCheckNewMsgStatus(hSession, ZFDIR_OUT, &fChanged) == 0 &&  fChanged)
       {
             /* call ZfxGetMsgListEx */
```

```
            ...
        }
        /* sleep for a few seconds */
}
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxCheckServer

Check server running.

Syntax

ZFERR FAR ZfxCheckServer(ZFSESSIONHANDLE hSession)

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call

Description

This routine checks whether the Zetafax server is running. The server must be running before messages can be submitted for sending.

Return value

The routine returns 0 if the server is running correctly, otherwise one of the following:

ZFERR_NOT_INITIALISED
ZFERR_SERVER_NOT_RUNNING

Example

```
#include  <stdio.h>
#include <zfapi.h>

 ...
 if (ZfxCheckServer(hSession) != 0)
 {
        printf("Problem checking Zetafax server\n");
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxCreateAutoFile

Creates file name with unique body.

Syntax

```
ZFERR FAR ZfxCreateAutoFile( ZFSESSIONHANDLE hSession, char FAR *lpszPrefix,  char FAR
*lpszExtn, char FAR *lpszPath, char FAR *lpszBody)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call lpszPrefix4 character identifier for filelpszExtn1 to 3 character extension for file
lpszPath	Full pathname of directory to contain filelpszBodyAddress of buffer (of length ZFMSG_BODY_LEN+1) to receive body name of file which is created (returned)

Description

This routine creates a file in the given directory, where no other file in the directory has the same body name (ie the part of the file name excluding the extension). The file name created has the form "~ppppnnn.xxx" where pppp and xxx are the prefix name and extension specified in the call, and nnn is a number from 000 to 999 (or can be 3 alphanumeric characters if the directory already contains a large number of matching files).

The routine is used when creating message files to be sent, since each message has a unique body name. It is recommended that the first character of the base name chosen is "X" to guarantee that files created by application programs are not confused with ones created by the Zetafax client (although this is not mandatory).

Return value

The routine returns 0 if successful, otherwise one of the following:

```
 ZFERR_NOT_INITIALISED
 ZFERR_INVALID_PARAMETERS
 ZFERR_PATH_NOT_FOUND
 ZFERR_TOO_MANY_FILES
 ZFERR_FILE_CREATE_ERROR
```

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
 char szBody[ZFMSG_BODY_LEN+1];
 char szUserOutDir[256];
if (ZfxGetUserOutDir(hSession, szUserOutDir,  sizeof(szUserOutDir)) == 0 &&
ZfxCreateAutoFile(hSession, "XSUB",  "TMP", szUserOutDir, szBody) == 0)
 {
```

```
      printf("Created file %s%s.TMP\n", szUserOutDir, szBody);
}
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxCreateCtlFile

Create a CONTROL file from SUBMIT format file.

Syntax

```
ZFERR FAR ZfxCreateCtlFile( ZFSESSIONHANDLE hSession, char FAR *lpszSubmitFile,  char FAR *lpszControlFile, char FAR *lpszBody, char FAR *lpszDataExtn, char FAR *lpszDataExtnBuf, char FAR *lpszCommentBuf)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call lpsz
SubmitFile	Name of submit file (with path if not in current directory)
lpszControl	FileName of new CONTROL file (with path if not in current directory)
lpszBody	Data file extension for the required format, or NULL to use the default (or to allow the Syntax line in the SUBMIT file to overwrite the default).
lpszDataExtn	lpszDataExtnBufAddress of buffer of length 4 bytes. On return this will contain the data file extension to use. If the lpszDataExtn parameter was given as NULL then this will give the data format specified in the Syntax line in the SUBMIT file (or the default value of TXT for straight ASCII text if this line is not specified). If the lpszDataExtn was specified then an error is returned if the SUBMIT file contains a Syntax line which contradicts this.
lpszCommentBuf	Address of buffer of length ZFMSG_COMMENT_LEN+1 used to return the message comment for specifying in a subsequent ZfxSendMsg call.

Description

This routine interprets the %%[MESSAGE] of a given SUBMIT format file, writing the details to the given CONTROL file. It is used where the ZfxSendSubmitFile function does not give sufficient flexibility - for example when wishing to send an existing data file.

NOTE - this function is supplied for compatibility with version 5 of the Zetafax API only. Version 6 API applications should use ZfxCreateCtlFileEx.

Return value

The routine returns 0 if successful, otherwise one of the following:

```
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_FILE_ERROR
ZFERR_FILE_OPEN_ERROR
ZFERR_FILE_CREATE_ERROR
ZFERR_SUBMIT_FILE_INVALID
```

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
/* create .CTL file in user's OUT directory */
 ZfxGetUserOutDir(hSession, szOutDir, sizeof(szOutDir);
 ZfxCreateAutoFile(hSession, "XSUB", "CTL", szOutDir, szBody); continued.
/* get name of the CTL file we've just created  */
 sprintf(szPath, "%s%s.CTL", szOutDir, szBody);
/* interpret the SUBMIT file created earlier */
 /* to send an existing file of format G3F */
 ZfxCreateCtlFile(hSession, "MYFILE.SUB", szPath, szBody, "G3F", szDataExtn,
szComment);
/* Create data file in OUT directory */
 sprintf(szPath, "%s%s.G3F", szOutDir, szBody);
 ...
/* submit files */
 ZfxSendMsg(hSession, szBody, "G3F", szComment);
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxCreateCtlFileEx

Create a CONTROL file from SUBMIT format file.

Syntax

```
ZFERR FAR ZfxCreateCtlFileEx( ZFSESSIONHANDLE hSession, char FAR  *lpszSubmitFile, char FAR *lpszControlFile, char FAR *lpszDataExtn, char FAR  *lpszDataExtnBuf, short MsgInfoExSize, ZFMSGINFOEX FAR *lpMsgInfoEx)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call lpszSubmitFileName of submit file (with path if not in current directory)
lpszControl	FileName of new CONTROL file (with path if not in current directory)lpszDataExtnData file extension for the required format, or NULL to use the default (or to allow the Syntax line in the SUBMIT file to overwrite the default).
lpszDataExtnBuf	Address of buffer of length 4 bytes. On return this will contain the data file extension to use. If the lpszDataExtn parameter was given as NULL then this will give the data format specified in the Syntax line in the SUBMIT file (or the default value of TXT for straight ASCII text if this line is not specified). If the lpszDataExtn was specified then an error is returned if the SUBMIT file contains a Syntax line which contradicts this.
MsgInfoExSize	Size of structure - should be set to sizeof(ZFMSGINFOEX) lpMsgInfoEx Address of a single ZFMSGINFOEX structure, which is filled with the details of the message (body, comment etc) on return.

Description

This routine interprets the %%[MESSAGE] of a given SUBMIT format file, writing the details to the given CONTROL file. It is used where the ZfxSendSubmitFile function does not give sufficient flexibility - for example when wishing to send an existing data file.

Return value

The routine returns 0 if successful, otherwise one of the following:

ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_FILE_ERROR
ZFERR_FILE_OPEN_ERROR
ZFERR_FILE_CREATE_ERROR
ZFERR_SUBMIT_FILE_INVALID

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...

/* create .CTL file in user's OUT directory  */

ZfxGetUserOutDir(hSession, szOutDir, sizeof(szOutDir));

ZfxCreateAutoFile(hSession, "XSUB", "CTL", szOutDir, MsgInfoEx.szBody);

/* get name of the CTL file we've just  created */

sprintf(szPath, "%s%s.CTL", szOutDir, MsgInfoEx.szBody);

/* interpret the SUBMIT file created earlier  */
/* to send an existing file of format G3F */

ZfxCreateCtlFileEx(hSession, "MYFILE.SUB", szPath, "G3F", szDataExtn,  sizeof
(ZFMSGINFOEX), &MsgInfoEx);

/* Create data file in OUT directory */

sprintf(szPath, "%s%s.G3F", szOutDir, MsgInfoEx.szBody);
...

/* submit files */

ZfxSendMsgEx(hSession, "G3F", sizeof(ZFMSGINFOEX), &MsgInfoEx);
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxCreateCtlFileFP

Create a CONTROL file (old version).

Syntax

ZFERR FAR ZfxCreateCtlFileFP(ZFSESSIONHANDLE hSession, FILE *fpSubmit, FILE *fpControl, char FAR *lpszBody, char FAR *lpszDataExtn, char FAR *lpszDataExtnBuf, char FAR *lpszCommentBuf)

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call fpSubmit File pointer for SUBMIT format file, opened in binary mode with read access (eg an "rb" parameter to the fopen call). The file pointer should be positioned at the start of the %%[MESSAGE] line. On return the file pointer is positioned at the start of the next line found in the file which starts with "% %" (signifying the end of the %%MESSAGE section) if one exists, or the end of file otherwise.fpControl File pointer for new CONTROL file, opened with in binary mode with write access.
lpszBody	Message body name (base name of control and data file) lpszDataExtn-Data file extension for the required format, or NULL to use the default (or to allow the Syntax line in the SUBMIT file to overwrite the default).
lpszDataExtnBuf	Address of buffer of length 4 bytes. On return this will contain the data file extension to use. If the lpszDataExtn parameter was given as NULL then this will give the data format specified in the Syntax line in the SUBMIT file (or the default value of TXT for straight ASCII text if this line is not specified). If the lpszDataExtn was specified then an error is returned if the SUBMIT file contains a Syntax line which contradicts this.
lpszComment	Buf Address of buffer of length ZFMSG_COMMENT_LEN+1

used to return the message comment for specifying in a subsequent ZfxSendMsg call.

Description

This routine interprets the %%[MESSAGE] of a given SUBMIT format file, writing the details to the given CONTROL file. It is used where the ZfxSendSubmitFile function does not give sufficient flexibility - for example when wishing to send an existing data file.
NOTE - this routine is supplied for historic reasons to assist porting applications written for an older version of the API. Because of its use of file pointers is only supported in the static library version of the API, not the DLL. New programs should use the ZfxCreateCtlFileEx function instead.

Return value

The routine returns 0 if successful, otherwise one of the following:

ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_FILE_ERROR
ZFERR_SUBMIT_FILE_INVALID

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
/* create .CTL file in user's OUT directory */
ZfxGetUserOutDir(hSession, szOutDir, sizeof(szOutDir);
ZfxCreateAutoFile(hSession, "XSUB", "CTL", szOutDir, szBody);

/* open the CTL file we've just created */
sprintf(szPath, "%s%s.CTL", szOutDir, szBody);
fpCtl = fopen(szPath, "wb");

/* interpret the SUBMIT file created earlier */
/* to send an existing G3F format file */
ZfxCreateCtlFileFP(hSession, fpSubmit, fpCtl, szBody, "G3F", szDataExtn,  szComment);
fclose(fpCtl);

/* Create data file in OUT directory */
sprintf(szPath, "%s%s.G3F", szOutDir, szBody);
 ...

/* submit files */
ZfxSendMsg(hSession, szBody, "G3F", szComment);
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxCreateDataFile

Create a DATA file from SUBMIT format file.

Syntax

```
ZFERR FAR ZfxCreateDataFile( ZFSESSIONHANDLE hSession, char FAR  *lpszSubmitFile, char FAR *lpszDataFile, char FAR *lpszDataExtn)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call
lpszSubmitFile	Name of submit file (with path if not in current directory) lpszDataFileName of new DATA file (with path if not in current directory) lpszDataExtnData file extension for the required format - either "TXT" for ASCII text, or "EPN" for Epson print format.

Description

This routine interprets the %%[TEXT] of a given SUBMIT format file, writing the details to the given CONTROL file. It is used where the ZfxSendSubmitFile function does not give sufficient flexibility - for example when wishing to send an existing data file.

Return value

The routine returns 0 if successful, otherwise one of the following:
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_FILE_OPEN_ERROR
ZFERR_FILE_CREATE_ERROR
ZFERR_FILE_ERROR
ZFERR_SUBMIT_FILE_INVALID

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
/* create .CTL file in user's OUT directory */
 ...
/* Create data file in OUT directory */
 sprintf(szPath, "%s%s.EPN", szOutDir, szBody);
 ZfxCreateDataFile(hSession, "MYFILE.SUB", szPath, "EPN");
/* submit files */
 ...
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxCreateDataFileFP

Create a DATA file (old version).

Syntax

```
ZFERR FAR ZfxCreateDataFile( ZFSESSIONHANDLE hSession, FILE *fpSubmit, FILE  *fpData,
char FAR *lpszDataExtn)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call
fpSubmit	File pointer for SUBMIT format file, opened in binary mode with read access (eg an "rb" parameter to the fopen call). The file pointer should be positioned at the start of the %%[TEXT] line. The routine interprets all the file from the current position to the end of the file, treating the contents as the message to be sent.
fpData	File pointer for new DATA file, opened in binary mode with write access.
lpszDataExtn	Data file extension for the required format - either "TXT" for ASCII text, or "EPN" for Epson print format.

Description

This routine interprets the %%[TEXT] of a given SUBMIT format file, writing the details to the given CONTROL file. It is used where the ZfxSendSubmitFile function does not give sufficient flexibility - for example when wishing to send an existing data file.
NOTE - this routine is supplied for historic reasons to assist porting applications written for an older version of the API. Because of its use of file pointers is only supported in the static library version of the API, not the DLL. New programs should use the ZfxCreateDataFile function instead.

Return Value

The routine returns 0 if successful, otherwise one of the following:

```
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_FILE_ERROR
ZFERR_SUBMIT_FILE_INVALID
```

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
```

```
/* create .CTL file in user's OUT directory */
 ...

/* create data file */
sprintf(szPath, "%s%s.EPN", szOutDir, szBody);
fpData = fopen(szPath, "wb");
ZfxCreateDataFileFP(hSession, fpSubmit, fpData,  "EPN");
fclose(fpData);

/* submit files */
 ...
```

Related topics

Alphabetical reference
Function error returns and reference

ZfxDeleteMsg

Delete message entry.

Syntax

ZFERR FAR ZfxDeleteMsg(ZFSESSIONHANDLE hSession, ZFMSGDIR MsgDir, char FAR *lpszBody, short fDeleteFiles)

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPI
Init call MsgDir	Message type - ZFDIR_OUT for sent messages, or ZFDIR_IN for received messages
lpszBody	Message body name - NULL to delete all messages for this user fDeleteFilesBoolean, non-zero if the control and data files (and any other temporary files created by the server) for the message are to be deleted, in addition to removing the message entry from the INFO file.

Description

This routine is called to remove the entry for a given message from the OUT or IN directory message INFO file, and optionally to delete the associated data files. If the server is stopped and the body name is specified as NULL then this routine will reset the specified directory (and its subdirectories) to delete all entries from the queue. Note that if the fDeleteFiles flag is set in this case all files will be deleted from the OUT directory and its subdirectories or the Z-IN directory, and a new empty INFO file generated.

Return value

The routine returns 0 if successful, otherwise one of the following:
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_SERVER_NOT_RUNNING
ZFERR_UNKNOWN_MESSAGE
ZFERR_INFO_FILE_OPEN_ERROR
ZFERR_INFO_FILE_ERROR
ZFERR_INFO_FILE_INVALID
ZFERR_MESSAGE_NOT_COMPLETED
ZFERR_SERVER_RUNNING

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
```

```
/* following call fails if message status not */
/* ZFMSG_FAILED or ZFMSG_OK */
if (ZfxDeleteMsg(hSession, ZFDIR_OUT, "~XSND000",  TRUE) == 0)
 {
        printf("Message deleted\n");
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxGetAPIVersion

Get version identifier for API routines.

Syntax

```
ZFERR FAR ZfxGetAPIVersion( char FAR *lpszBuffer, short BufLen, unsigned short  FAR
*lpusVersion)
```

Parameters

Parameter	Description
lpszBuffer	Address of buffer to store version identifier string, or NULL if not required. BufLenLength of buffer (including space for terminating null)
lpusVersion	Address of unsigned short integer variable used to return the version number of the API routines, or NULL if not required.

Description

This routine gets a version identifier for the API routines. The version identifier is 20 characters or fewer (generally about 5 characters plus terminating null), and may be used when displaying the version number an application program.

The numeric version number will allow future programs to detect when they are running with an older version of the API. This is currently set to 0x600 (hex) - it is recommended that programs do not object if this number is higher than expected to allow for future updates to the API without the need to rebuild the programs.

Return value

The routine returns 0 if successful, otherwise one of the following:
ZFERR_BUFFER_TOO_SMALL

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...

char szVersion[20+1];
unsigned short usVersion;
printf("Automatic invoicing program v2.27\n");

if (ZfxGetAPIVersion(szVersion, sizeof(szVersion),  &usVersion) == 0)
 {
        printf("Zetafax API version %s\n", szVersion);
 }
```

Related topics
Alphabetical reference

Function error returns and reference

ZfxGetMsgDefaultsEx

Gets the default message settings for the user.

Syntax

ZFERR FAR ZfxGetMsgDefaultsEx(ZFSESSIONHANDLE hSession, short MsgDefaultsExSize, ZFMSGDEFAULTSEX *lpMsgDefaultsEx)

Parameters

Parameter	Description
Hsession	API session handle, as returned by ZfxAPIInit call
MsgDefaultsExSize	Size of ZFMSGDEFAULTSEX structure. Must be the size (in bytes) of a ZFMSGDEFAULTSEX structure
lpMsgDefaultsEx	Pointer to memory allocated for a ZFMSGDEFAULTSEX structure

Description

This routine is called to get the default message settings for the Zetafax user associated with the API session associsted with the passed session handle.

Return value

The routine returns 0 if successful, otherwise one of the following:
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
 ZFMSGDEFAULTSEX ZfMsgDefaults;
 memset(&ZfMsgDefaults, (int) 0, sizeof(ZFMSGDEFAULTSEX)
 if (ZfxGetMsgDefaultsEx(hSession, sizeof(ZFMSGDEFAULTSEX), &ZfMsgDefaults) == 0)
 {
        /* 'ZfMsgDefaults' structure now */
       /* contains the default message */
       /* settings for the user */
       printf("Default message priority is %d\n", ZfMsgDefaults.Priority); ...
 }
```

Related topics
Alphabetical reference
Function error returns and reference
ZFGetMSGDEFAULTSEX structure

ZfxGetMsgHistory

Get transmission history information for message.

Syntax

```
ZFERR FAR ZfxGetMsgHistory( ZFSESSIONHANDLE hSession, ZFMSGDIR MsgDir, char FAR
*lpszBody, short AddrNum, short fAllEvents, short MaxEntries, short FAR  *lpNumEntries,
short MsgHistorySize, ZFMSGINFO FAR *lpMsgHistory)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call MsgDirMessage type - ZFDIR_OUT for sent messages, or ZFDIR_IN for received messages
lpszbody	Message file body AddrNumWhich addressee events are required for (first addressee is number 1), or 0 to retrieve events for all addressees.
fAllEvents	Non zero if events of type ZFEVENT_TRIED are to be included, otherwise only "final state" events are included.
MaxEntries	Maximum number of entries to be returned (ie number of elements in the arrays which follow)
lpNumEntries	Address of short integer variable used to return the number of events in the file. Note that the returned value may be larger than the value given for MaxEntries if the buffer was not large enough for all entries.
MsgHistorySize	Size of structure - should be set to sizeof (ZFMSGHISTORY) lpMsgHistoryAddress of array of 'n' ZFMSGHISTORY structures, where 'n' is the value given by the MaxEntries parameter

Description

This routine gets a list of the transmission history entries in the CONTROL file for a given message, giving the information about transmission attempts (result, connection time etc).

If the number of entries in the list exceeds the MaxEntries parameter, then the routine returns information about the first MaxEntries entries, but sets the lpNumEntries parameter to the total number of entries in the list. Be careful therefore to only read the lesser of (MaxEntries) and (lpNumEntries) elements of the array on return. Calling the routine with MaxEntries set to 0 will just return a count of the number of entries.

The fAllEvents flag can be used to specify whether all dial and transmission attempts should be included, or just the final status. If only the final status for a given addressee is required then the function can be called with the fAllEvents flag set to FALSE (0), and a single ZFMSGHISTORY buffer (MaxEntries = 1).
The routine opens and reads the CONTROL file for the message. This is the file that is updated by the Zetafax server while the message is in its queue. It is therefore necessary to protect against calling this routine too frequently while the entry is in the server queue. Use ZfxCheckNewMsgStatus and ZfxGetMsgInfo to wait until the message has completed before calling this routine.

NOTE - this function is supplied for compatibility with version 5 of the Zetafax API only. Version 6 API applications should use ZfxGetMsgHistoryEx .

Return value

The routine returns 0 if successful, otherwise one of the following:
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_CONTROL_FILE_OPEN_ERROR
ZFERR_CONTROL_FILE_ERROR
ZFERR_CONTROL_FILE_INVALID

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
#define LIST_SIZE 20

ZFMSGHISTORY aMsgHistory[LIST_SIZE];
Err = ZfxGetMsgHistory(hSession, ZFDIR_OUT,  "~SEND000", 1, TRUE, LIST_SIZE,
&NumEntries, sizeof(ZFMSGHISTORY),  MsgHistory);
if (Err != 0)
 {
       return;
 }
if (NumEntries > LIST_SIZE)
 {
      printf("Total messages %d\n",  NumEntries); printf("Displaying first %d\n",
LIST_SIZE);
 }
for (Entry = 0, Tries = 1; Entry <  min(NumEntries, LIST_SIZE); Entry++)
 {
      switch(aMsgHistory[Entry]->EventType)
      {

            case ZFEVENT_TRIED: Tries++;
                  printf("Tried  unsuccessfully\n");
            break;

            case ZFEVENT_SENT_OK:
                  Tries++;
                  printf("Sent  OK on attempt %d\n" Tries);
                  printf("Connect  time %d secs\n", MsgHistory[Event].ConnectSecs);
            break;

            case ZFEVENT_SENT_ERROR:
                  Tries++;
```

```
                printf("Failed  after %d attempts\n", Tries);
                printf("Pages  sent %d\n", aMsgHistory[Event].Pages);
            break;

            default:
            /*  ignore other events */
            break;
        }
}
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxGetMsgHistoryEx

Get transmission history information for message.

Syntax

```
ZFERR FAR ZfxGetMsgHistoryEx( ZFSESSIONHANDLE hSession, ZFMSGDIR MsgDir, char  FAR
*lpszBody, short AddrNum, SHORT fAllEvents, short MaxEntries, short FAR  *lpNumEntries,
short MsgHistoryExSize, ZFMSGHISTORYEX FAR *lpMsgHistoryEx)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call MsgDirMessage type - ZFDIR_OUT for sent messages, or ZFDIR_IN for received messages
lpszbody	Body of message file to get history forAddrNumWhich addressee events are required for (first addressee is number 1), or 0 to retrieve events for all addressees.
fAllEvents	Non zero if events of type ZFEVENT_TRIED are to be included, otherwise only "final state" events are included. MaxEntriesMaximum number of entries to be returned (ie number of elements in the arrays which follow)
lpNumEntries	Address of short integer variable used to return the number of events in the file. Note that the returned value may be larger than the value given for MaxEntries if the buffer was not large enough for all entries.
MsgHistoryExSize	Size of structure - should be set to sizeof (ZFMSGHISTORYEX) lpMsgHistoryExAddress of array of 'n' ZFMSGHISTORYEX structures, where 'n' is the value given by the MaxEntries parameter

Description

This routine gets a list of the transmission history entries in the CONTROL file for a given message, giving the information about transmission attempts (result, connection time etc).

If the number of entries in the list exceeds the MaxEntries parameter, then the routine returns information about the first MaxEntries entries, but sets the lpNumEntries parameter to the total number of entries in

the list. Be careful therefore to only read the lesser of (MaxEntries) and (lpNumEntries) elements of the array on return. Calling the routine with MaxEntries set to 0 will just return a count of the number of entries.

The fAllEvents flag can be used to specify whether all dial and transmission attempts should be included, or just the final status. If only the final status for a given addressee is required then the function can be called with the fAllEvents flag set to FALSE (0), and a single ZFMSGHISTORY buffer (MaxEntries = 1).

The routine opens and reads the CONTROL file for the message. This is the file which is updated by the Zetafax server while the message is in its queue. It is therefore necessary to protect against calling this routine too frequently while the entry is in the server queue - this can be done by using the ZfxCheckNewMsgStatus and ZfxGetMsgInfoEx routines to wait until the message has completed before calling this routine.

Return value

The routine returns 0 if successful, otherwise one of the following:
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_CONTROL_FILE_OPEN_ERROR
ZFERR_CONTROL_FILE_ERROR
ZFERR_CONTROL_FILE_INVALID

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
#define LIST_SIZE 20

ZFMSGHISTORYEX aMsgHistoryEx[LIST_SIZE];
Err = ZfxGetMsgHistoryEx(hSession, ZFDIR_OUT,  "~SEND000", 1, TRUE, LIST_SIZE,
&NumEntries, sizeof(ZFMSGHISTORYEX),  MsgHistoryEx);

if (Err != 0)
{
       return;
 }

if (NumEntries  > LIST_SIZE)
 {
       printf("Total  messages %d\n", NumEntries);
       printf("Displaying first %d\n",  LIST_SIZE);
 }

for (Entry = 0, Tries = 1; Entry <  min(NumEntries, LIST_SIZE); Entry++)
 {
        switch(aMsgHistoryEx[Entry]->EventType)
       {
              case ZFEVENT_TRIED:
                     Tries++;
                     printf("Tried  unsuccessfully\n");
              break;

              case ZFEVENT_SENT_OK:
                     Tries++;
                     printf("Sent  OK on attempt %d\n", Tries);
                     printf("Connect  time %d secs\n", aMsgHistory[Event].ConnectSecs);
              break;

              case ZFEVENT_SENT_ERROR:
                     Tries++;
```

```
                printf("Failed  after %d attempts\n", Tries);
                printf("Pages  sent %d\n", aMsgHistory[Event].Pages);
            break;

            default: /* ignore other events */
            break;
        }
}
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxGetMsgInfo

Get information about single message.

Syntax

```
ZFERR FAR ZfxGetMsgInfo( ZFSESSIONHANDLE hSession, ZFMSGDIR MsgDir, char FAR  *lpszBody,
ZFMSGINFO FAR *lpMsgInfo)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call MsgDirMessage type - ZFDIR_OUT for sent messages, or ZFDIR_IN for received messages
lpszBody	Message body name - NULL to delete all messages for this user lpMsgInfoAddress of a single ZFMSGINFO structure, which is filled with the details of the message (status etc) on return.

Description

This routine gets information about a message in the user's OUT or IN directories. Note that if the status of several messages is required the ZfxGetMsgList function should be used instead, as this is more efficient than several calls to this routine.

NOTE - this function is supplied for compatibility with version 5 of the Zetafax API only. Version 6 API applications should use ZfxGetMsgInfoEx .

Return value

The routine returns 0 if successful, otherwise one of the following:

ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_INFO_OPEN_FILE_ERROR
ZFERR_INFO_FILE_ERROR
ZFERR_INFO_FILE_INVALID
ZFERR_UNKNOWN_MESSAGE

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
ZFMSGINFO MsgInfo;
if (ZfxGetMsgInfo(hSession, ZFDIR_OUT, "~XSND000",  &MsgInfo) == 0
&& (MsgInfo.Status == ZFMSG_OK ||  MsgInfo.Status == ZFMSG_FAILED)
{
```

```
      /*  ok to delete message */
}
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxGetMsgInfoEx

Get information about single message.

Syntax

```
ZFERR FAR ZfxGetMsgInfoEx( ZFSESSIONHANDLE hSession, ZFMSGDIR MsgDir, char FAR
*lpszBody, short MsgInfoExSize, ZFMSGINFOEX FAR *lpMsgInfoEx)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call MsgDirMessage type - ZFDIR_OUT for sent messages, or ZFDIR_IN for received messages
lpszBody	Body of message file to retrieve MsgInfoExSizeSize of the ZFMSGINFOEX structure. Should be set to sizeof (ZFMSGINFOEX)
lpMsgInfoEx	Address of a single ZFMSGINFOEX structure, which is filled with the details of the message (status, comment, subject etc) on return.

Description

This routine gets information about a message in the user's OUT or IN directories. Note that if the status of several messages is required the ZfxGetMsgListEx function should be used instead, as this is more efficient than several calls to this routine.

Return value

The routine returns 0 if successful, otherwise one of the following:
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_INFO_OPEN_FILE_ERROR
ZFERR_INFO_FILE_ERROR
ZFERR_INFO_FILE_INVALID
ZFERR_UNKNOWN_MESSAGE

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
ZFMSGINFOEX MsgInfoEx;
if (ZfxGetMsgInfoEx(hSession, ZFDIR_OUT, "~XSND000",  sizeof(ZFMSGINFOEX), &MsgInfoEx)
== 0
 && (MsgInfoEx.Status == ZFMSG_OK || MsgInfoEx.Status == ZFMSG_FAILED))
{
```

```
        /* ok to delete message */
}
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxGetMsgList

Get status of all messages for user.

Syntax

```
ZFERR FAR ZfxGetMsgList( ZFSESSIONHANDLE hSession, ZFMSGDIR MsgDir, short  MaxEntries,
short FAR *lpNumEntries, ZFMSGINFO FAR *lpMsgInfo)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call MsgDirMessage type - ZFDIR_OUT for sent messages, or ZFDIR_IN for received messages
MaxEntries	Maximum number of entries to be returned (ie number of elements in the arrays which follow) lpNumEntriesAddress of short integer variable used to return the number of events in the file. Note that the returned value may be larger than the value given for MaxEntries if the buffer was not large enough for all entries.
lpMsgInfo	Address of array of 'n' ZFMSGINFO structures, where 'n' is the value given by the MaxEntries parameter.

Description

This routine gets a list of the entries in the OUT or IN window list for this user, together with information about each one (current status, comment etc). If the program wants to find the status of just one message it is more efficient to call the ZfxGetMsgInfo routine.

If the number of entries in the list exceeds the MaxEntries parameter, then the routine returns information about the first MaxEntries entries, but sets the lpNumEntries parameter to the total number of entries in the list. Be careful therefore to only read the lesser of (MaxEntries) and (lpNumEntries) elements of the array on return. Calling the routine with MaxEntries set to 0 will just return a count of the number of entries.

Entries are added by selecting Send (File menu) in the Zetafax client, by calling the ZfxSendMsgEx routine, or when a new message is received. Entries are removed by selecting Save (File menu) or Delete (File menu) in the Zetafax client or calling ZfxDeleteMsg. A message may also be removed when it completes if it has the "delete on completion" option set, or for an incoming message if it is routed to another user or to an e-mail InBox.

The ZfxCheckNewMsgStatus routine should be used in conjunction with this call if repeatedly checking the status of messages, to prevent reading the INFO file unnecessarily.

NOTE - this function is supplied for compatibility with version 5 of the Zetafax API only. Version 6 API applications should use ZfxGetMsgListEx .

Return value

The routine returns 0 if successful, otherwise one of the following:
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_INFO_FILE_OPEN_ERROR
ZFERR_INFO_FILE_ERROR
ZFERR_INFO_FILE_INVALID

Example

```
#include <stdio.h>
#include <zfapi.h>
 ...
#define LIST_SIZE 20

ZFMSGINFO aMsgInfo[LIST_SIZE];
Err = ZfxGetMsgList(hSession, ZFDIR_OUT, LIST_SIZE,  &NumEntries, aMsgInfo);

if (Err != 0)
{
       return;
}

if (NumEntries > LIST_SIZE)
{
       printf("Total messages %d\n",  NumEntries);
       printf("Displaying first %d\n",  LIST_SIZE);
}

for (Entry = 0; Entry < min(NumEntries, LIST_SIZE); Entry++)
 {
       printf("Entry %d : %s, status %d\n",  Entry, MsgInfo[Entry].szBody, MsgInfo
[Entry].Status);
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxGetMsgListEx

Get status of all messages for user.

Syntax

```
ZFERR FAR ZfxGetMsgListEx( ZFSESSIONHANDLE hSession, ZFMSGDIR MsgDir, short  MaxEntries,
short FAR *lpNumEntries, short MsgInfoExSize, ZFMSGINFOEX FAR  *lpMsgInfoEx)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call MsgDirMessage type - ZFDIR_OUT for sent messages, or ZFDIR_IN for received messages
MaxEntries	Maximum number of entries to be returned (ie number of elements in the arrays which follow) lpNumEntriesAddress of short integer variable used to return the number of events in the file. Note that the returned value may be larger than the value given for MaxEntries if the buffer was not large enough for all entries.
MsgInfoExSize	Size of the ZFMSGINFOEX structure, as given by sizeof (ZFMSGINFOEX).
lpMsgInfoEx	Address of array of 'n' ZFMSGINFOEX structures, where 'n' is the value given by the MaxEntries parameter.

Description

This routine gets a list of the entries in the OUT or IN window list for this user, together with information about each one (current status, comment, subject etc). If the program wants to find the status of just one message it is more efficient to call the ZfxGetMsgInfoEx routine.

If the number of entries in the list exceeds the MaxEntries parameter, then the routine returns information about the first MaxEntries entries, but sets the lpNumEntries parameter to the total number of entries in the list. Be careful therefore to only read the lesser of (MaxEntries) and (lpNumEntries) elements of the array on return. Calling the routine with MaxEntries set to 0 will just return a count of the number of entries.

Entries are added by selecting Send (File menu) in the Zetafax client, by calling the ZfxSendMsgEx routine, or when a new message is received. Entries are removed by selecting Save (File menu) or Delete (File menu) in the Zetafax client or calling ZfxDeleteMsg. A message may also be removed when it completes if it has the "delete on completion" option set, or for an incoming message if it is routed to another user or to an e-mail InBox.

The ZfxCheckNewMsgStatus routine should be used in conjunction with this call if repeatedly checking the status of messages, to prevent reading the INFO file unnecessarily.

Return value

The routine returns 0 if successful, otherwise one of the following:
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_INFO_FILE_OPEN_ERROR
ZFERR_INFO_FILE_ERROR
ZFERR_INFO_FILE_INVALID

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
#define LIST_SIZE 20

ZFMSGINFOEX aMsgInfoEx[LIST_SIZE];
Err = ZfxGetMsgListEx(hSession, ZFDIR_OUT,  LIST_SIZE, &NumEntries, sizeof(ZFMSGINFOEX),
aMsgInfoEx);

if (Err != 0)
 {
       return;
 }

if (NumEntries > LIST_SIZE)
 {
       printf("Total messages %d\n",  NumEntries);
       printf("Displaying first %d\n",  LIST_SIZE);
 }

for (Entry = 0; Entry < min(NumEntries,  LIST_SIZE); Entry++)
 {
       printf("Entry %d : %s, status %d\n",  Entry, MsgInfoEx[Entry].szBody, MsgInfoEx
[Entry].Status);
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxGetServerStatus

Get information about the server.

Syntax

```
ZFERR FAR ZfxGetServerStatus( ZFSESSIONHANDLE hSession, short ServerInfoSize,
ZFSERVERINFO FAR *lpServerInfo short MaxDevices, short FAR *lpNumDevices, short
DeviceInfoSize, ZFDEVICEINFO FAR *lpDeviceInfo)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call ServerInfoSizeSize of structure - should be set to size of (ZFSERVERINFO) lpServerInfoAddress of a ZFSERVERINFO structure used to return status information for the server
MaxDevices	Maximum number of device entries to be returned (ie number of elements in the DeviceInfo array)lpNumDevicesAddress of short integer variable used to return the number of events in the file. Note that the returned value may be larger than the value given for MaxDevices if the buffer was not large enough for all entries.
DeviceInfoSize	Size of structure - should be set to sizeof (ZFDEVICEINFO)
lpDeviceInfo	Address of array of 'n' ZFDEVICEINFO structures, where 'n' is the value given by the MaxDevices parameter.

Description

This routine gets the current status of the server (number of entries in queues etc), and of each device.

If the number of devices configured exceeds the MaxDevices parameter, then the routine returns information about the first MaxDevices devices, but sets the lpNumDevices parameter to the total number of devices in the list. Be careful therefore to only read the lesser of (MaxDevices) and (lpNumDevices) elements of the array on return. Calling the routine with MaxDevices set to 0 will just return a count of the number of devices.

NOTE - this function is supplied for compatibility with version 5 of the Zetafax API only. Version 6 API applications should use ZfxGetServerStatusEx.

Return value

The routine returns 0 if successful, otherwise one of the following:
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_SERVER_NOT_RUNNING
ZFERR_FILE_OPEN_ERROR
ZFERR_FILE_ERROR
ZFERR_CANT_SUBMIT_REQUEST

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
#define LIST_SIZE 20

ZFSERVERINFO ServerInfo;
ZFDEVICEINFO aDeviceInfo[LIST_SIZE];
Err = ZfGetServerStatus(hSession,  sizeof(ZFSERVERINFO), &ServerInfo,
LIST_SIZE, &NumDevices,  sizeof(ZFDEVICEINFO), aDeviceInfo);

if (Err == 0)
 {
       printf("Items waiting for device = %d\n",  ServerInfo.QueueWaitingDevice);

       if (NumDevices >= 1 &&  aDeviceInfo[0].szUser[0] != '\0')
       {
              printf("First  device is processing %s:%s", aDeviceInfo[0].szUser,
aDeviceInfo[0].szMsgBody);
       }
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxGetServerStatusEx

Get information about the server.

Syntax

```
ZFERR FAR ZfxGetServerStatusEx( ZFSESSIONHANDLE hSession, short  ServerStatusExSize,
ZFSERVERSTATUSEX FAR *lpServerStatusEx)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call
ServerStatusExSizeSize	of structure - should be set to size of (ZFSERVERSTATUSEX)
lpServerStatusEx	Address of a ZFSERVERSTATUSEX structure used to return status information for the server

Description

This routine gets the current status of the server (number of entries in queues etc), of each device, and of each link to a remote server.
If the number of devices configured exceeds the lpServerStatusEx->MaxDevices parameter, then theroutine returns information about the firstlpServerStatusEx->MaxDevices devices, but sets the lpServerStatusEx->lpNumDevices parameter to the total number of devices in the list. Be careful therefore to onlyread the lesserof (lpServerStatusEx->MaxDevices) and (lpServerStatusEx->lpNumDevices) elements of the array on return. Calling theroutine with lpServerStatusEx->MaxDevices set to 0 will just return a count of the number of devices. The same logic also applies to the useof lpServerStatusEx->MaxLinks andlpServerStatusEx->lpNumLinks.

Return value

The routine returns 0 if successful, otherwise one of the following:
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_SERVER_NOT_RUNNING
ZFERR_FILE_OPEN_ERROR
ZFERR_FILE_ERROR
ZFERR_CANT_SUBMIT_REQUEST

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
#define DEV_LIST_SIZE 20
#define LINK_LIST_SIZE 30

ZFSERVERSTATUSEX ServerStatus;
ZFSERVERINFOEX ServerInfo;
ZFDEVICEINFOEX aDeviceInfo[DEV_LIST_SIZE];
ZFLINKINFOEX aLinkInfo[LINK_LIST_SIZE];
```

```
short NumDevices;
short NumLinks;

/* initialise required fields */
memset(&ServerStatus, 0, sizeof(ServerStatus));
ServerStatus.ServerInfoExSize =  sizeof(ZFSERVERINFOEX);
ServerStatus.lpServerInfoEx = &ServerInfo;
ServerStatus.MaxDevices = DEV_LIST_SIZE;
ServerStatus.lpNumDevices = &NumDevices;
ServerStatus.DeviceInfoExSize = sizeof(ZFDEVICEINFOEX);
ServerStatus.lpDeviceInfoEx = aDeviceInfo;
ServerStatus.MaxLinks = LINK_LIST_SIZE;
ServerStatus.lpNumLinks = &NumLinks;
ServerStatus.LinkInfoExSize = sizeof(ZFLINKINFOEX);
ServerStatus.lpLinkInfoEx = aLinkInfo;

Err = ZfxGetServerStatusEx(hSession,  sizeof(ZFSERVERSTATUSEX), &ServerStatus);

if (Err == 0)
 {
        printf("Items waiting for device =  %d\n", ServerInfoEx.QueueWaitingDevice);

       if (NumDevices >= 1 &&  aDeviceInfo[0].szUser[0] != '\0')
       {
              printf("Device  1 processing %s:%s", aDeviceInfo[0].szUser, aDeviceInfo[0
].szMsgBody);
       }

       if (NumLinks >= 1)
       {
       printf("%d  faxes sent OK via '%s'", aLinkInfo[0].NumSentOK, aLinkInfo[0].
szRemoteServer);
       }
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxGetSystemArea

Get location of system file directory.

Syntax

```
ZFERR FAR ZfxGetSystemArea( ZFSESSIONHANDLE hSession, char FAR *lpszBuffer,  short BufLen)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit
calllpszBufferAddress of	buffer used to return the full path of the directory, including a terminating backslash ('\') and NULL.
BufLen	Size of the buffer (including space for the terminating NULL)

Description

This routine returns the full path name of the base directory used for storing system shared files, with a backslash added to the end (eg "S:\ZFAX\SYSTEM\"). The terminating backslash means that a valid path can be formed by appending a file name to this string. This should not normally be required by API programs.

Return value

The routine returns 0 if successful, otherwise one of the following:
ZFERR_NOT_INITIALISED
ZFERR_BUFFER_TOO_SMALL

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
szPath[256];
if (ZfxGetSystemArea(hSession, szPath,  sizeof(szPath)) == 0) { ... }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxGetUserArea

Get location of user's base directory.

Syntax

ZFERR FAR ZfxGetUserArea(ZFSESSIONHANDLE hSession, char FAR *lpszBuffer, short BufLen)

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit
calllpszBufferAddress of	buffer used to return the full path of the directory, including a terminating backslash ('\') and NULL.
BufLen	Size of the buffer (including space for the terminating NULL)

Description

This routine returns the full path name of the user's base directory , with a backslash added to the end (eg "S:\ZFAX\USERS\FRED\"). The terminating backslash means that a valid path can be formed by appending a file name to this string. The directory should not normally be required by the API.

Return value

The routine returns 0 if successful, otherwise one of the following:

ZFERR_NOT_INITIALISED
ZFERR_BUFFER_TOO_SMALL

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
szPath[256];
if (ZfxGetUserArea(hSession, szPath,  sizeof(szPath)) == 0) { ... }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxGetUserCoversheet

Get user's default coversheet.

Syntax

ZFERR FAR ZfxGetUserCoversheet(ZFSESSIONHANDLE hSession, char FAR *lpszBuffer, short BufLen)

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit calllpszBufferAddress of buffer used to return the full path of the directory, including a terminating backslash ('\') and NULL.
BufLen	Size of the buffer (including space for the terminating NULL)

Description

Each user in Zetafax can set their own default setting for the coversheet they want to use. This routine returns the name of their default coversheet, or a null string if their default setting is for no coversheet to be sent.

The routine need only be called if the program wants to perform specific processing on the coversheet name (for example, overriding it if blank), as the default will automatically be used if the "Coversheet:" line is omitted from a SUBMIT file.

The maximum length of coversheet names equal to ZFMSG_COVERSHEET_LEN (excluding space for the terminating NULL).

Return value

The routine returns 0 if successful, otherwise one of the following:

ZFERR_NOT_INITIALISED
ZFERR_BUFFER_TOO_SMALL

Example

```
#include <stdio.h>
#include <zfapi.h>
 ...

szCoversheet[ZFMSG_COVERSHEET_LEN+1];

if (ZfxGetUserCoversheet(hSession, szCoversheet, sizeof(szCoversheet)) != 0
 || szCoversheet[0] == '\0') /*if  default is no coversheet */
 {
       /* force a coversheet to be used */
       strcpy(szCoversheet, "COVSHEET");
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxGetUserFromname

Get user's default sender name.

Syntax

```
ZFERR FAR ZfxGetUserFromname( ZFSESSIONHANDLE hSession, char FAR *lpszBuffer,  short BufLen)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit calllpszBufferAddress of buffer used to return the full path of the directory, including a terminating backslash ('\') and NULL.
BufLen	Size of the buffer (including space for the terminating NULL)

Description

This routine returns the default setting for the sender name for messages submitted by this user (usually their full name). When a new user is created the sender name is set to the full name entered in the Zetafax Configuration program, but the user can then change it using the Zetafax client. Note that editing an existing user in the Setup program does not change the sender name.
The routine need only be called if the program wants to perform specific processing on the name (for example, overriding it if blank), as the default will automatically be used if the "From:" line is omitted from a SUBMIT file.
Sender names have a maximum length of ZFMSG_FULLNAME_LEN (excluding space for the terminating NULL).

Return Value

The routine returns 0 if successful, otherwise one of the following:

ZFERR_NOT_INITIALISED
ZFERR_BUFFER_TOO_SMALL

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
szFromname[ZFMSG_FULLNAME_LEN+1];
if (ZfxGetUserFromname(hSession, szFromname,  sizeof(szFromname)) != 0
       || szFromname[0] =='\0') /* if no default  name set */
 {
       /* force a default name */
       strcpy(szFromname, "Fax Administrator");
 }
```

Related topics

Alphabetical reference
Function error returns and reference

ZfxGetUserInDir

Get location of user's IN directory.

Syntax

```
ZFERR FAR ZfxGetUserInDir( ZFSESSIONHANDLE hSession, char FAR *lpszBuffer, short
BufLen)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call
lpszBuffer	Address of buffer used to return the full path of the directory, including a terminating backslash ('\') and NULL. BufLen
	Size of the buffer (including space for the terminating NULL)

Description

This routine returns the full path name of the user's OUT directory, with a backslash added (eg "S:\ZFAX\USERS\FRED\Z-OUT\"). The terminating backslash means that a valid path can be formed by appending a file name to this string.

Return value

The routine returns 0 if successful, otherwise one of the following:
ZFERR_NOT_INITIALISED
ZFERR_BUFFER_TOO_SMALL

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...

szPath[256];
if (ZfxGetUserInDir(hSession, szPath,  sizeof(szPath)) == 0)
 {
       ...
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxHoldMsg

Request server to suspend processing of message.

Syntax

```
ZFERR FAR ZfxHoldMsg( ZFSESSIONHANDLE hSession, ZFMSGDIR MsgDir, char FAR  *lpszBody)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPI
Init call	MsgDirMessage type (ZFDIR_OUT)
lpszBody	Base name of message file

Description

This routine is called to stop the server processing a message after it has been submitted using ZfxSendMsgEx. The routine sends a request to the Zetafax server, which processes it asynchronously.

Held messages remain in the queue in their original position until released or deleted. Any transmissions in progress when the hold request is received will not be interrupted, but no further preparation will take place and the message will not be submitted to any new devices for sending. Normal processing is resumed by calling the ZfxReleaseMsg function.

Return value

The routine returns 0 if successful, otherwise one of the following:

```
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_UNKNOWN_MESSAGE
ZFERR_SERVER_NOT_RUNNING
ZFERR_CANNOT_SUBMIT_REQUEST
```

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...

if (ZfxHoldMsg(hSession, ZFDIR_OUT, "~XSND000") ==  0)
 {
       printf("Message hold request sent\n");
       /* Processing not stopped until status */
       /* changes to ZFMSG_HELD */
       ...
 }
```

Related topics

Alphabetical reference
Function error returns and reference

ZfxMarkMsgAsRead

Marks message as 'read' (changes the user status of the message).

Syntax

```
ZFERR FAR ZfxMarkMsgAsRead( ZFSESSIONHANDLE hSession, ZFMSGDIR MsgDir, char FAR
*lpszBody)
```

Parameters

Parameter	Description
Hsession	API session handle, as returned by ZfxAPIInit call MsgDir Message type (ZFDIR_OUT)LpszBodyBase name of message file

Description

This routine is called to mark a specified message as 'read'. It updates the user staus of the message in the control files so does not need to send any requests to the Zetafax Server.

Return value

The routine returns 0 if successful, otherwise one of the following:

```
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_UNKNOWN_MESSAGE
ZFERR_SERVER_NOT_RUNNING
```

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
 if (ZfxMarkMsgAsRead(hSession, ZFDIR_OUT, "~XSND000") == 0)
 {
        printf("Message marked as read\n");
        /* viewing the user's message */
        /*in the client would now show */
        /* the message as being read */
        ...
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxReleaseMsg

Request server to resume processing of message.

Syntax

ZFERR FAR ZfxReleaseMsg(SESSIONHANDLE hSession, ZFMSGDIR MsgDir, char FAR *lpszBody)

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit cal
lMsgDir	Message type (ZFDIR_OUT)
lpszBody	Base name of message file

Description

This routine is called to resume the server processing a message that has been suspended with ZfxHoldMsg . The routine sends a request to the Zetafax server, which processes it asynchronously.

Return value

The routine returns 0 if successful, otherwise one of the following:

ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_UNKNOWN_MESSAGE
ZFERR_SERVER_NOT_RUNNING
ZFERR_CANNOT_SUBMIT_REQUEST

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...

if (ZfxReleaseMsg(hSession, ZFDIR_OUT, "~XSND000")  == 0)
 {
       printf("Message release request sent\n");
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxRestartServer

Restart the Zetafax server.

Syntax

```
ZFERR FAR ZfxRestartServer( ZFSESSIONHANDLE hSession, SHORT fReserved)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by by ZfxAPIInit call
fReserved	Reserved - set to FALSE (0)

Description

This routine requests the Zetafax server to restart if already running. This is equivalent to ZfxStopServer followed by ZfxStartServer, but can be called from any machine (unlike ZfxStartServer which must be called on the fax server PC).

When the server programs restart, they re-read the initialization files and settings. ZfxRestartServer can therefore be used after making changes to the server settings, such as adding or removing fax devices. Note however that restarting the server will abort any faxes currently being sent or received.

Return value

The routine returns 0 if successful, otherwise one of the following:

```
ZFERR_NOT_INITIALISED
ZFERR_SERVER_NOT_RUNNING
```

Example

```
#include <stdio.h>
#include <zfapi.h>

 ...

Err = ZfxRestartServer(hSession, FALSE);

if (Err == 0)
 {
       printf("Server restarting\n");  }
 else
 {
       printf("Unable to restart server\n");
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxRushMsg and ZfxRushMsgEx

Request server to stop processing message.

Syntax

```
ZFERR FAR ZfxRushMsg( ZFSESSIONHANDLE hSession, ZFMSGDIR MsgDir, char FAR  *lpszBody)
ZFERR FAR ZfxRushMsgEx( ZFSESSIONHANDLE hSession, ZFMSGDIR MsgDir, char FAR  *lpszBody,
char FAR *lpszMsgID)
```

Parameters

Parameter	Description
Hsession	API session handle, as returned by ZfxAPIInit call MsgDir Message type (ZFDIR_OUT)
lpszBody	Base name of message file
lpszMsgID	Message ID

Description

These routines are called to request the server rush the processing a message after it has been submitted using ZfxSendMsgEx. The routine sends a request to the Zetafax server, which processes it asynchronously.

Return value

The routine returns 0 if successful, otherwise one of the following:
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_UNKNOWN_MESSAGE
ZFERR_SERVER_NOT_RUNNING
ZFERR_CANNOT_SUBMIT_REQUEST

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...

if (ZfxRushMsg(hSession, ZFDIR_OUT, "~XSND000") == 0)
 {
        printf("Message rushed");
        /* server should now speed up the */
        /* processing of the message */
        /* now wait for status to change to */
        /* ZFMSG_OK or ZFMSG_FAILED before */
        /* deleting the message */
        ...
 }
```

Related topics
Alphabetical reference

Function error returns and reference

ZfxSendMsg

Submit message to fax server for sending.

Syntax

```
ZFERR FAR ZfxSendMsg( ZFSESSIONHANDLE hSession, char FAR *lpszBody, char FAR
*lpszDataExtn, char FAR *lpszComment)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call
lpszBody	Body name of control file and data filelpszDataExtnExtension of data file corresponding to the data format
lpszComment	Descriptive comment (of maximum length ZFMSG_COMMENT_LENGTH). This is the comment that appears on the Zetafax client display OUT directory list.

Description

This routine makes an entry for a new message in the user's OUT directory INFO file then sends a message to the Zetafax server asking it to queue the message for sending.

The caller should first create two files in the users OUT directory (as given by the ZfxGetUserOutDir function). These are the control file (with extension ".CTL"), and the data file (with an extension corresponding to the data file format, as specified in File System Structure section above. The two files should have the same body name, which will usually have been determined by calling the ZfxCreateAutoFileroutine.

Note that once a message has been submitted to the fax server for processing, the server will periodically open the control file to change the status lines or append message history and status information to the file. It is recommended that application programs avoid opening the control file until the status of the message changes to ZFMSG_OK or ZFMSG_FAILED, at which point the server has completed processing of the message. However, if the program requires to read the file during this time then it must not be kept open for more than 2 seconds, otherwise status updates may be lost.

NOTE - this function is supplied for compatibility with version 5 of the Zetafax API only. Version 6 API applications should use ZfxSendMsgEx .

Return value

The routine returns 0 if successful, otherwise one of the following:

```
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_SERVER_NOT_RUNNING
ZFERR_FILE_NOT_FOUND
ZFERR_MESSAGE_ALREADY_EXISTS
```

ZFERR_INFO_FILE_OPEN_ERROR
ZFERR_INFO_FILE_ERROR
ZFERR_INFO_FILE_INVALID
ZFERR_CANNOT_SUBMIT_REQUEST

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...

 /* create ~XSND000.CTL and ~XSND000.TXT */
 ...

if (ZfxSendMsg(hSession, "~XSND000", "TXT", "To: Sam  Smith") == 0)
 {
       printf("Submitted to Zetafax server\n");}
 else
 {
       /* delete the two files */
       ...
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxSendMsgEx

Submit message to fax server for sending.

Syntax

```
ZFERR FAR ZfxSendMsgEx( ZFSESSIONHANDLE hSession, char FAR *lpszDataExtn, short
MsgInfoExSize, ZFMSGINFOEX lpMsgInfoEx)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call
lpszDataExtn	Extension of data file corresponding to the data format
MsgInfoExSize	Size of the ZFMSGINFOEX structure, as returned by sizeof(ZFMSGINFOEX).
lpMsgInfoEx	Address of a single ZFMSGINFOEX structure, containing the body, comment and subject of the message to be submitted.

Description

This routine makes an entry for a new message in the user's OUT directory INFO file then sends a message to the Zetafax server asking it to queue the message for sending.

The caller should first create two files in the users OUT directory (as given by the ZfxGetUserOutDir function). These are the control file (with extension ".CTL"), and the data file (with an extension corresponding to the data file format, as specified in File System Structure section above. The two files should have the same body name, which will usually have been determined by calling the ZfxCreateAutoFileroutine.

Note that once a message has been submitted to the fax server for processing, the server will periodically open the control file to change the status lines or append message history and status information to the file. It is recommended that application programs avoid opening the control file until the status of the message changes to ZFMSG_OK or ZFMSG_FAILED, at which point the server has completed processing of the message. However, if the program requires to read the file during this time then it must not be kept open for more than 2 seconds, otherwise status updates may be lost.

Return value

The routine returns 0 if successful, otherwise one of the following:

```
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_SERVER_NOT_RUNNING
ZFERR_FILE_NOT_FOUND
ZFERR_MESSAGE_ALREADY_EXISTS
ZFERR_INFO_FILE_OPEN_ERROR
ZFERR_INFO_FILE_ERROR
ZFERR_INFO_FILE_INVALID
```

ZFERR_CANNOT_SUBMIT_REQUEST

Example

```
#include <stdio.h>
#include <zfapi.h>
 ...
ZFMSGINFOEX MsgInfo;
 ...
 /* create ~XSND000.CTL and ~XSND000.TXT */
 ...
 if (ZfxSendMsgEx(hSession, "TXT", sizeof(ZFMSGINFOEX), &MsgInfo) == 0)
 {
       f("Submitted to Zetafax server\n");}
 else
 {
       /* delete the two files */
       ...
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxSendSubmitFile

Send a single SUBMIT format file.

Syntax

```
ZFERR FAR ZfxSendSubmitFile( ZFSESSIONHANDLE hSession, char FAR  *lpszSubmitFile, char FAR *lpszPrefix, char FAR *lpszBody)
```

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call
lpszSubmitFile	Name of submit file (with path if not in current directory)lpszPrefix4 characters string giving the prefix to use when creating the control and data files.
lpszBody	Address of buffer of length ZFMSG_BODY_LEN+1 used to return the message body name if successfully submitted for sending.

Description

This routine interprets the given SUBMIT format file, creating a CONTROL file and a DATA file (in ASCII text or Epson print format). It then calls the ZfxSendMsgEx function to submit these files for sending, and if successful returns the name of the message submitted. This is the simplest method of submitting a fax using the API.

Return value

The routine returns 0 if successful, otherwise one of the following:

```
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_FILE_OPEN_ERROR
ZFERR_FILE_ERROR
ZFERR_MESSAGE_ALREADY_EXISTS
ZFERR_SERVER_NOT_RUNNING
ZFERR_SUBMIT_FILE_INVALID
ZFERR_INFO_FILE_OPEN_ERROR
ZFERR_INFO_FILE_ERROR
ZFERR_INFO_FILE_INVALID
ZFERR_CANNOT_SUBMIT_REQUEST
```

Example

```
#include  <stdio.h>
#include <zfapi.h>
...

if ((fp = fopen("XYZ.TMP", "w+b")) != NULL)
{
```

```
        fputs("%%[MESSAGE]\r\n", fp);
        fputs("From: Fred Smith\r\n", fp);
        fputs("To: Jim Jones\r\n", fp);
        fputs("Fax: 123 456 7890\r\n", fp);
        fputs("%%[TEXT]\r\n", fp);
        fputs("Hello Jim\r\n", fp);
        fclose(fp);
        Err = ZfxSendSubmitFile(hSession, "XYZ.TMP", "XSUB",  szBody);
        remove("XYZ.TMP");
}
if (Err == 0)
 {
        printf("Submitted message %s\n", szBody);
 }
```

Related topics

Alphabetical reference
Function error returns and reference

ZfxSendSubmitFileFP

Send a single SUBMIT format file (old version).

Syntax

ZFERR FAR ZfxSendSubmitFileFP(ZFSESSIONHANDLE hSession, FILE *fpSubmit, char FAR *lpszPrefix, char FAR *lpszBody)

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call
fpSubmit	File pointer for SUBMIT format file, opened in binary mode with read access (eg an "rb" parameter to the fopen call). The file pointer should be positioned at the start of the %%[MESSAGE] line.
lpszPrefix	4 characters string giving the prefix to use when creating the control and data files.
lpszBody	Address of buffer of length ZFMSG_BODY_LEN+1 used to return the message body name if successfully submitted for sending.

Description

This routine interprets the given SUBMIT format file, creating a CONTROL file and a DATA file (in ASCII text or Epson print format). It then calls the ZfxSendMsgEx function to submit these files for sending, and if successful returns the name of the message submitted. This is the simplest method of submitting a fax using the API.

The SUBMIT file must have been opened with read access - ie if the file is created by the application program first, then it should be opened with access "w+" not just "w".

NOTE - this routine is supplied for historic reasons to assist porting applications written for an older version of the API. Because of its use of file pointers is only supported in the static library version of the API, not the DLL. New programs should use the ZfxSendSubmitFile function instead.

Return value

The routine returns 0 if successful, otherwise one of the following:

```
ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_FILE_ERROR
ZFERR_MESSAGE_ALREADY_EXISTS
ZFERR_SERVER_NOT_RUNNING
ZFERR_SUBMIT_FILE_INVALID
ZFERR_INFO_FILE_OPEN_ERROR
```

ZFERR_INFO_FILE_ERROR
ZFERR_INFO_FILE_INVALID
ZFERR_CANNOT_SUBMIT_REQUEST

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...

if ((fp = fopen("XYZ.TMP", "w+b")) != NULL)
{
       fputs("%%[MESSAGE]\r\n", fp);
       fputs("From: Fred Smith\r\n", fp);
       fputs("To: Jim Jones\r\n", fp);
       fputs("Fax: 123 456 7890\r\n", fp);
       fputs("%%[TEXT]\r\n", fp);
       fputs("Hello Jim\r\n", fp);
       fseek(fp, 0L, SEEK_SET);
       Err = ZfxSendSubmitFileFP(hSession, fp, "XSUB",  szBody);
       fclose(fp);
       remove("XYZ.TMP");
}

if (Err == 0)
 {
        printf("Submitted message %s\n", szBody);
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxStartServer

Start the Zetafax server on this PC.

Syntax

ZFERR FAR ZfxStartServer(ZFSESSIONHANDLE hSession, SHORT fReserved, char FAR *lpszReserved)

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPIInit call
fReserved	Reserved - set to FALSE (0)
lpszReserved	Reserved - set to NULL

Description

This routine starts the Zetafax server on the current PC. The PC must have been correctly configured to run the server. The server programs are started in turn. This routine returns once the server has begun starting, but the calling application should then use the ZfxCheckServer routine to determine when it has started.

Return value

The routine returns 0 if successful, otherwise one of the following:

ZFERR_NOT_INITIALISED
ZFERR_INVALID_PARAMETERS
ZFERR_SERVER_RUNNING
ZFERR_CANNOT_RUN_SYSTEM_MANAGER
ZFERR_ERROR_STARTING_SERVER

Example

```
#include <stdio.h>
#include <zfapi.h>
 ...

Err = ZfxStartServer(hSession, FALSE, NULL);

if (Err == 0)
 {
       printf("Server starting\n");
 }
 else
 {
       printf("Unable to start server\n");
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxStopServer

Stop the Zetafax server.

Syntax

ZFERR FAR ZfxStopServer(ZFSESSIONHANDLE hSession, SHORT fReserved)

Parameters

Parameter	Description
hSession	API session handle, as returned by ZfxAPI
Init callfReserved	Reserved - set to FALSE (0)

Description

This routine requests the Zetafax server to stop. Note that this will abort any faxes currently being sent or received.

Return value

The routine returns 0 if successful, otherwise one of the following:
ZFERR_NOT_INITIALISED
ZFERR_SERVER_NOT_RUNNING

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...
Err = ZfxStopServer(hSession, FALSE);

if (Err == 0)
 {
       printf("Server stopping\n");
 }

else
 {
       printf("Unable to stop server\n");
 }
```

Related topics
Alphabetical reference
Function error returns and reference

ZfxVBSendSubmitFile

Send a single SUBMIT format file.

Syntax

ZFERR FAR ZfxVBSendSubmitFile(char FAR *lpszUsername, char FAR *lpszSubmitFile, char FAR *lpszPrefix)

Parameters

Parameter	Description
lpszUserName	Zetafax username to use for submitting the message
lpszSubmitFile	Full path and file name of SUBMIT format file to send.
lpszPrefix	4 characters string giving the prefix to use when creating the control and data files.

Description

This routine interprets the given SUBMIT format file, creating a CONTROL file and a DATA file (in ASCII text or Epson print format). It then calls the ZfxSendMsgEx function to submit these files for sending.

This is equivalent to calling ZfxAPIInit, ZfxSendSubmitFile and ZfxAPIClosedown. It is supplied for programs that can call functions in the DLL, but are unable to store the session handle returned by ZfxAPIInit and pass it to the other two functions - word processor macro languages and some Visual Basic applications, for example.

The ZfxAPIInit function has to do a certain amount of work to determine the system configuration and user settings, so it is more efficient when submitting multiple faxes to keep a session open (calling ZfxAPIInit once when the application starts for example) then repeatedly call ZfxSendSubmitFile. It is therefore recommended that this function is only used by programs that are unable to use ZfxSendSubmitFile .

Return value

The routine returns 0 if successful, otherwise one of the following:

ZFERR_INVALID_PARAMETERS
ZFERR_NO_ZF_INIT_FILE
ZFERR_INVALID_ZF_INIT_FILE
ZFERR_UNKNOWN_USER
ZFERR_CANNOT_LOG_ON
ZFERR_FILE_OPEN_ERROR
ZFERR_FILE_ERROR ZFERR_SERVER_NOT_RUNNING
ZFERR_SUBMIT_FILE_INVALID
ZFERR_INFO_FILE_OPEN_ERROR
ZFERR_INFO_FILE_ERROR
ZFERR_INFO_FILE_INVALID
ZFERR_CANNOT_SUBMIT_REQUEST

Example

```
#include  <stdio.h>
#include <zfapi.h>
 ...

if ((fp = fopen("XYZ.TMP", "w+b")) != NULL)
{
       fputs("%%[MESSAGE]\r\n", fp);
       fputs("From: Fred Smith\r\n", fp);
       fputs("To: Jim Jones\r\n", fp);
       fputs("Fax: 123 456 7890\r\n", fp);
       fputs("%%[TEXT]\r\n", fp);
       fputs("Hello Jim\r\n", fp);
       fclose(fp);
       Err = ZfxVBSendSubmitFile("FRED", "XYZ.TMP",  "XSUB");
       remove("XYZ.TMP");

       if (Err == 0)
       {
              printf("Submitted message\n");
       }
}
```

Related topics
Alphabetical reference
Function error returns and reference

DDE commands

Applications such as word processors may communicate with the Zetafax client program directly using DDE commands. Provided adequate details have first been given using the DDE commands, when a document is "printed" using the Zetafax Windows printer driver no dialogs will be displayed and the fax or faxes will be submitted to the Zetafax server automatically.

DDE conversation

Before issuing any addressing commands, a DDE conversation must be established between the application and the Zetafax client program. The name of the DDE server is **Zetafax**, and the topic for the purposes of addressing faxes is **Addressing**.

The Zetafax client program should be put under the control of DDE by issuing a DDEControl DDE Execute call.

Addressing commands can then be issued using DDE Poke calls. Details of the commands which may be used are given below.

A message file should be created by printing using the Zetafax Windows printer driver. Alternatively an ASCII file, or a suitable Epson or TIFF file, may be copied to the spool file location used by the Zetafax client program (usually C:\WINDOWS\ZETAFAX.SPL, set by the LogArea: entry in the ZETAFAX.INI file).

The message can be submitted to the Zetafax server for sending with a Send DDE Execute call, or for preview with a Preview DDE Execute call.

Further messages can be submitted by specifying a new addressee, organization, and fax number, creating a new message file, and then issuing a Send or Preview DDE Execute call.

Note: Each of the addressing settings, such as the choice of letterhead, will remain unchanged between messages until the appropriate DDE Poke call is made to alter it. If these settings need to be reset to their default values between messages, the Zetafax client program should be released from DDE control with a DDERelease DDE Execute call and then put back under DDE control with a DDEControl DDE Execute call.

After submitting the message or messages the Zetafax client program should be released from the control of DDE by issuing a DDERelease DDE Execute call.

The DDE conversation should finally be terminated properly.

Commands

The following WM_DDE_EXECUTE commands may be issued using the Addressing topic:

Command	Description
DDEControl	Puts the Zetafax client program under DDE control for addressing.
Send	Submits a message to the Zetafax server for sending.
Preview	Submits a message to the Zetafax server for preview.
DDERelease	Releases the Zetafax client from DDE control.

Poke items

The following table lists the items that may be poked (WM_DDE_POKE) using the Addressing topic whilst the Zetafax client program is under DDE control on all Zetafax configurations:

Item	Parameters
To	fax, recipient name, organization

Fax	fax
Name	recipient name
Organization	organization

For a description of each command and its parameters see SUBMIT file Message Addressing lines.

With an API license the following additional items may be poked:

Item	Parameters
From	sender name
Coversheet	coversheet
Letterhead	letterhead
Quality	quality
Priority	priority
After	time
Time	time
Header	header
Attach	files
Charge	chargecode

For a full description of each command and its parameters, see SUBMIT file Message Option lines.

Related topics
Example DDE macros

Example DDE macros

Windows 98

The following Microsoft Word macro demonstrates the use of DDE commands to submit the current document to the Zetafax server for sending by fax:

```
Sub MAIN
REM Set up DDE control of Zetafax
Conv1 = DDEInitiate("Zetafax", "Addressing")
DDEExecute(Conv1, "[DDEControl]")

REM Set the addressing options
DDEPoke(Conv1, "To", "123 456 7890, Sam Smith, Smith and Sons")

REM Set Zetafax To the default printer And Print the document
FilePrintSetup .Printer = "Zetafax    printer on ZETAFAX.SPL"
FilePrint

REM Submit the fax And release DDE control
DDEExecute(Conv1, "[Send][DDERelease]")
DDETerminate(Conv1)
End Sub
```

Background printing

Please note that this macro disables the background printing feature when printing to the Zetafax printer from within a Microsoft Word macro. This is done by adding the following line:

ToolsOptionsPrint .Background = 0

You can also re-activate background printing by adding the following line to the end of your macro:

ToolsOptionsPrint .Background = 1

Windows NT

Because Windows NT uses a different naming convention for its printer ports it is necessary to adjust the macro shown above to accommodate for this. The simplest way to do this is to record a new macro where you select your Zetafax printer. In the example above, you need to change the FilePrintSetup call to specify the spool file path which would look something like:

FilePrintSetup .Printer = "Zetafax printer on NE00:"

This is best done by recording a simple macro within Word of you selecting the Zetafax printer and copying the resulting Word Basic code to your Zetafax DDE macro.

Here is an example Word for Windows macro which submits the current document to the fax server for sending by fax under Windows NT:

```
Sub MAIN
REM Set up DDE control of Zetafax
   Conv1 = DDEInitiate("Zetafax", "Addressing")
   DDEExecute(Conv1, "[DDEControl]")
REM Set Zetafax To the default printer And Print the document
   FilePrintSetup .Printer = "Zetafax printer on NE00:"
REM Disable background printing
```

```
   ToolsOptionsPrint .Background = 0
   FilePrint
REM Set the addressing options
   DDEPoke(Conv1, "To", "123 456 7890, Sam Smith, Smith & Sons")
   DDEPoke(Conv1, "Quality", "High")
   DDEPoke(Conv1, "Time", "19:00:00")
   DDEPoke(Conv1, "Attach", "infopack")
REM Submit the fax And release DDE    control
   DDEExecute(Conv1, "[Send][DDERelease]")
   DDETerminate(Conv1)
End Sub
```

Microsoft Excel

In Microsoft Excel, the DDEPoke instructions do not support a constant string as data. It must be a range address. So, instead of using the line:

```
DDEPoke Conv1, "From", "John Doe"
```

The instruction should appear as:

```
DDEPoke Conv1, "From", WorkSheets("Sheet1").Range("A1")
```

Here is an example Excel macro that will submit the current document to the fax server using addressing information contained in fields within an active worksheet.

```
Sub Macro1()
ZetaTalk = DDEInitiate( _ app:="Zetafax",_ topic:="Addressing")
   Application.ActivePrinter = "Zetafax printer on NE00:"
   ActiveSheet.PrintOut
   DDEExecute ZetaTalk, "[DDEControl]"
   Set RecipientName = Worksheets("Sheet1").Range("A1")
   Set OrgName = Worksheets("Sheet1").Range("B1")
   Set FaxNumber = Worksheets("Sheet1").Range("C1")
   Application.DDEPoke ZetaTalk, "Name", RecipientName
   Application.DDEPoke ZetaTalk, "Organisation", OrgName
   Application.DDEPoke ZetaTalk, "Fax", FaxNumber

Rem Submit the fax And release DDE control
   DDEExecute ZetaTalk, "[Send][DDERelease]"
   DDETerminate ZetaTalk
End Sub
```

Embedded Addressing

Like many fax packages, Zetafax has a Windows printer driver. Print from a Windows application, and a dialog box will pop-up asking you where the fax is to be sent. With the API this can be automated by including options such as the fax number in the document being printed. Zetafax will pick out the embedded addressing information and act upon it.

You can use embedded addressing to broadcast faxes from a database or using a word processor mailmerge so each recipient's copy will be personalised.

The Zetafax client program allows addressing instructions to be included into documents using embedded commands:

Embedded addressing information

The Zetafax client program allows addressing instructions to be embedded into documents using embedded commands.

Supported platforms

Embedded addressing is supported using the Zetafax Printer driver on the following operating systems:

- Windows XP
- Windows 2000
- Windows NT 4.0
- Windows 95/98

Embedded addressing is not currently supported on Microsoft Terminal Server, Windows Terminal Services or Citrix Metaframe environments.

Use without the API

Limited support for embedded addressing commands and DDE is standard in Zetafax. This guide gives details of all of the options available when the API toolkit is purchased for the product.

The concept

Word processor and other applications' documents may contain commands to indicate where a fax should be sent, together with a wide range of settings such as the time of sending, priority, resolution, coversheet and letterhead to use. If these commands are used in a document, they must be typed using one of Zetafax's own typefaces, and in the appropriate syntax - for example:

%%[Fax:123 456 7890]

When the document is "printed" using the Zetafax printer driver, the addressing commands are used by the Zetafax client program in place of the addressing dialog boxes. Provided adequate details are given, no dialog boxes will be displayed and the fax or faxes will be submitted to the fax server automatically.

Supported fonts

The Zetafax printer supports embedded addressing in Windows XP, Windows 2000, Windows NT 4.0 and Windows 95/98. When using embedded addressing commands they can be entered into a document in any available Windows font; however the embedded addressing commands must use the appropriate syntax and appear on a line of their own. Earlier versions of Zetafax under certain operating environments required the use of Zetafax specific fonts for embedded addressing.

Related topics
Options commands
Action commands
Using embedded addressing with mail merge

Commands

Option Commands

The addressing commands are similar in syntax to entries in Submit files. They are enclosed in the special characters %%[] to distinguish them from printable text. The embedded command text does not appear on the faxes produced by Zetafax.

The following commands are available on all Zetafax configurations.

To
Fax
Name
Organisation
Send
Preview

The remaining options commands are only available if the API toolkit is licensed:

From
Coversheet
Letterhead
Quality
Priority
Time
Header
Attach
Charge
Discard
Subject
Note
Delete

Related topics
Embedding addressing information
Using embedded addressing with mail merge

To

Syntax

%%[To: fax, recipientname, organisation]

where fax is the fax number to send the fax to, recipientname is the person the fax is addressed to, and organisation is that person's company or organisation; fax, recipientname, and organisation may each appear in double quotation marks if required.

Description

Specifies the fax number to use and the details of the recipient which are put on the fax cover sheet and at the top of each page of the fax.

Example

%%[To: 123 456 7890, Sam Smith, Smith and Sons]

Name

Syntax

%%[Name: recipientname]
where recipientname is the person the fax is addressed to.

Description

A subset of and alternative to the %%[To:]; used with the %%[Fax] command.

Example

%%[Name: Sam Smith]

Fax

Syntax

%%[Fax: fax]
where fax is the fax number to send the fax to.

Description

A subset of and alternative to the %%[To:] command.

Example

%%[Fax: 123 456 7890]

Organisation

Syntax

%%[Organisation: organisation]

where organisation is the recipient's company or organisation.

Description

A subset of and alternative to the %%[To:] command; used with the %%[Fax] command.

Example

%%[Organisation: Smith and Sons]

Send

Syntax

%%[SEND]

Description

Indicates that the document being printed should be split at the foot of this page and submitted as a fax. The remaining page or pages will be treated as a separate fax or faxes.

Example

%%[Send]

Preview

Syntax

%%[Preview]

Description

Indicates that the document being printed should be split at the foot of this page and submitted as a fax for preview. The remaining page or pages will be treated as a separate fax or faxes.

Example

%%[Preview]

From

Syntax

%%[FROM: sendername]

where sendername is the originator of the fax - it may appear in double quotation marks.

Description

Specifies the name which is put on the fax coversheet as the sender of the fax (cf the From: field on the Zetafax addressing dialog box).

Example

%%[From: Jim Jones]

Coversheet

Syntax

%%[COVERSHEET: coversheet]

where coversheet is the name of the file to be used to generate a coversheet for the fax (1 to 8 characters long) or blank for no coversheet - coversheet may appear in double quotation marks.

Description

Specifies what coversheet to generate for the fax onto before sending.

Example

%%[Coversheet: COVSHEET]

Quality

Syntax

%%[QUALITY: quality]

where quality is one of DRAFT, NORMAL or HIGH.

Description

Specifies the resolution to be used when sending the fax, in the same way as the Resolution button on the Zetafax client sending options dialog box. For fax, DRAFT and HIGH will normally force the fax to be sent at standard (200x100 dpi) and fine (200x200 dpi) resolutions respectively, whilst NORMAL will send at whichever resolution has been specified by the system administrator in the system initialization file.

Example

%%[Quality: NORMAL]

Letterhead

Syntax

%%[LETTERHEAD: letterhead]

where letterhead is the name of the file to be used as the letterhead (1 to 8 characters long) or blank for no letterhead - letterhead may appear in double quotation marks.

Description

Specifies what letterhead to merge the fax onto before sending.

Example

%%[Letterhead: LETTHEAD]

Priority

Syntax

%%[PRIORITY: priority]

where priority is one of URGENT, NORMAL or BACKGROUND.

Description

Specifies the priority to be used when queuing the fax, in the same way as the Priority radio button on the Zetafax client sending options dialog box.

Example

%%[Priority: NORMAL]

Time

Syntax

%%[AFTER: time] or %%[TIME: time]

where time specifies the earliest time that the message should be sent, for deferred sending. The format of the time field may be one of the following: hh:mm:ss (given time today) yy-mm-dd hh:mm:ss (date and time specified) OFFPEAK (as defined in SETUP.INI) If the time field is omitted the message is queued for sending immediately.

Description

Specifies when the message is to be sent.

Example

%%[After: 99-03-01 18:00:00]

Header

Syntax

%%[HEADER: header] where header is one or more of No, To, Fr, Dt and Ti.

Description

Specifies the header to appear at the top of each page of the fax, in the same way as the Header radio button on the Zetafax client sending options dialog box. No page numbering (n/N) To name of recipient Fr name of sender Da date Ti time

Example

%%[Header: NoToFrDt]

Attach

Syntax

%%[ATTACH: files]

where files is a comma separated list of graphics files to attach to the fax.

Description

Specifies the attachment files in the user's private graphics directory (Z-GRAPH) or in the system graphics directory, in the same way as the Attach radio button on the Zetafax client sending options dialog box.

Example

%%[Attach: INFOPACK, PRICES]

Charge

Syntax

%%[CHARGE: chargecode]

where chargecode is the charge code to be used for the fax

Description

Specifies the charge code to be used when queuing the fax, in the same way as the Charge combo box on the Zetafax client addressing dialog box.

Example

%%[Charge: SALES]

Discard

Syntax

%%[DISCARD]

Description

Specifies that this page will not be faxed. This command allows you to put all the embedded addressing commands on to one page and not have to worry about upsetting the formatting of the actual fax. Please note that this is not be available on Windows 95/98.

Example

%%[Discard]

Subject

Syntax

%%[SUBJECT: subjectline]

where subjectline is the subject of the fax.

Description

Specifies the subject field for the fax which can appear on the coversheet.

Example

%%[Subject: About the new sales figues]

Note

Syntax

%%[NOTE: notetext] where notetext is the text to appear in the coversheet note field.

Description

Specifies the note field text that can appear on the coversheet.

Example

%%[Note: Please find attached the price list]

Delete

Syntax

%%[DELETE: delete] where delete is YES, OK or NO.

Description

Specifies whether the fax should be deleted after sending. If delete is YES then the faxes are deleted after they have been sent (successful or failed). If delete is OK then the faxes are deleted after they have been sent successfully. If delete is NO then the faxes are **not** deleted after they have been sent (a blank line is equivalent).

Example

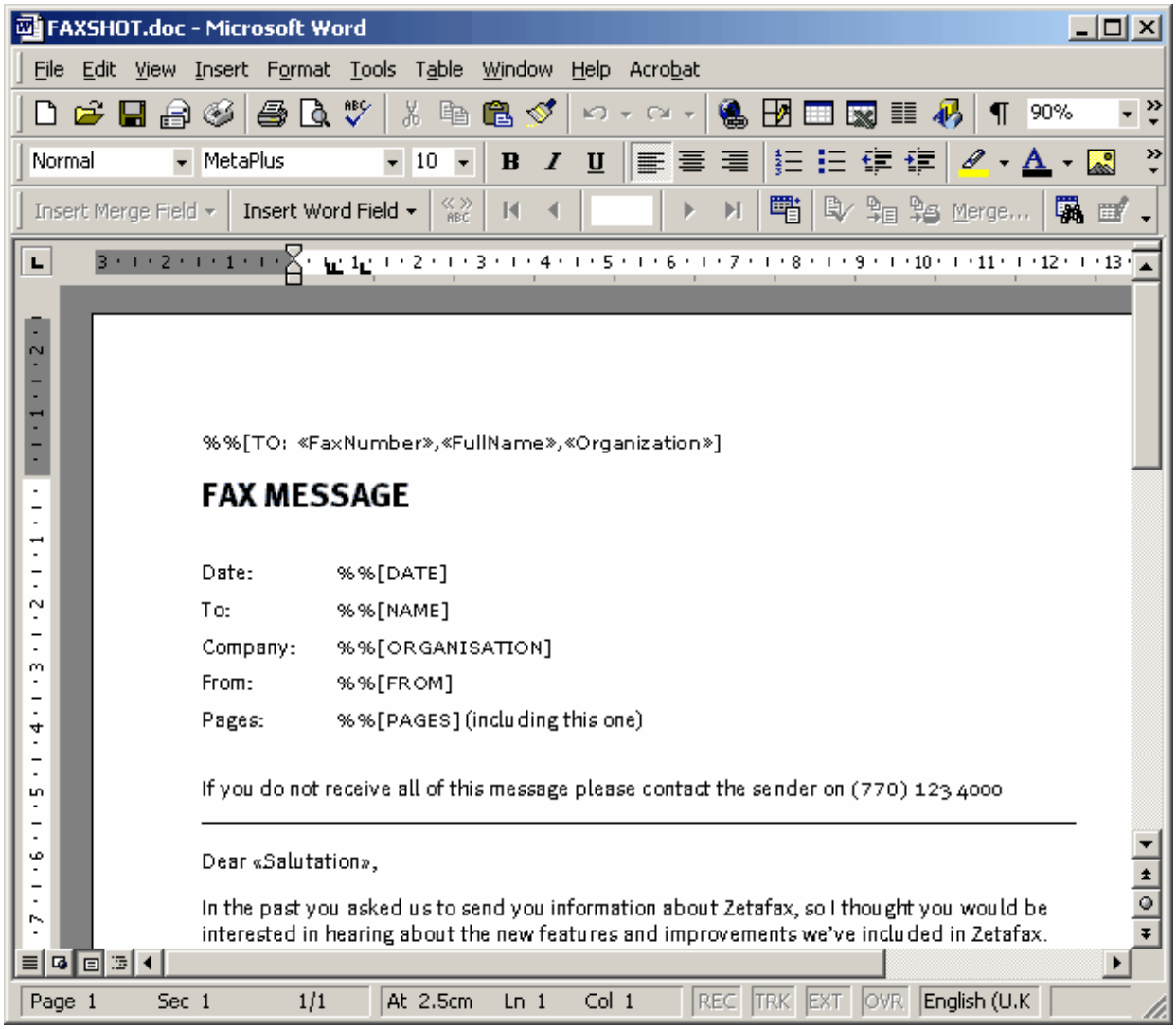
%%[Delete: YES]

Using embedded addressing with mail merge

Mail merge is used most frequently for merging mailing address details to a standard letter in order to send personalized copies to a list of recipients. The resulting documents are then printed to a network paper printer on company letterhead and mailed to the recipient. The embedded addressing feature of the Zetafax API can be used to send faxes to multiple recipients from a list generated by a contact manager or database.

Using Zetafax

In the same way, the addressing data fields can be replaced with fax addressing information by inserting mail merge fields in your word processing template enclosed within the Zetafax embedded addressing syntax. The addressing information is inserted when the mail merge is performed and instead of printing to a network paper printer, simply printing to the Zetafax printer renders the fax to the queue at the Zetafax server. At this stage, you can also tell Zetafax to overlay the document to your company's letterhead. For example, a section in a Microsoft Word template may look something like:



Related topics
Embedding addressing information
Options commands
Action commands

The ZSUBMIT program

The ZSUBMIT program is a part of the API toolkit which lets DOS, mini and mainframe programs send faxes and text messages by creating text files in a shared directory on a file server. For example, faxing purchase orders or invoices usually only requires adjusting an accounts packages report writer.
A text file needs a few lines added to give the fax/mobile number and message options. Logos and signatures can be included and the text may be merged onto a multi-page form using the letterhead feature in Zetafax. This approach gives results rapidly.

This section introduces and explains in depth the format of submit files required by the ZSUBMIT program. Typically, this section can be used as a reference for both users of ZSUBMIT and the COM and C language APIs. Finally, there are some example SUBMIT files giving instances of its use.

Creating SUBMIT files

One of the API methods of submitting messages for sending is to create a file in a specific format, termed SUBMIT file format. This is an ASCII text file that usually contains details of the options to use in preparing the message, the recipients of the message, as well as the message text itself. ZSUBMIT files have the .sub file extension.

ZSUBMIT program

The message may then be sent using the ZSUBMIT program. This program may be installed to run continuously as one of the Zetafax server programs, regularly checking a particular directory for the existence of a SUBMIT format file. If one is found then it is interpreted and submitted to the Zetafax server for sending - its progress may then be monitored using the client program as normal.

The ZSUBMIT program submits all messages as a given Zetafax user, and if required will monitor progress of the messages, limiting the number of outstanding messages at any one time or removing any entries which have been sent without errors. Files for sending via the ZSUBMIT program may also contain more than one message for sending - the program will split it into separate messages before submitting it to the server.

Use with API calls

SUBMIT format files are also used by the Application Programmers Interface (API) routines to specify message options and text for sending. These routines allow user written application programs to send messages directly without using either the client or ZSUBMIT program. The ZSUBMIT program itself is written using these routines, and requires the API toolkit license to run.

SUBMIT file format

This section details the format of SUBMIT files, for use with both the ZSUBMIT program and with the API routines. It gives the structure and fields allowed in the file for specifying message options and recipients, and also for formatting the message text.

The SUBMIT file is a standard ASCII text file (multiple lines, each terminated with CR LF). The file contains information about the options to be used in preparing a given message, the recipients, and the message text itself. The format is:

```
%%[MESSAGE]
Message option lines
Message addressing lines

%%[TEXT]
Text including message text fields
```

To specify a separate file containing the message to be sent (e.g. to send a graphics file):

```
%%[MESSAGE]
Message option lines
Message addressing lines

%%[FILE]
Filename, including path
```

Note: The first line of the file must be "%%[MESSAGE]".

The section headings ("%%[MESSAGE]" and "%%[TEXT]" or "%%[FILE]"), all message option and

addressing lines, and the filename (if the second form is used) must start at the left of the line (i.e. there must be no spaces preceding them). Finally, tab characters should not be used in the file - they will be replaced with a space character.

Multiple messages

Files submitted using the ZSUBMIT program may contain information about one or more messages by simply concatenating the files (i.e. the %%[MESSAGE] section for the second message follows the end of the %%[TEXT] or %%[FILE] sections for the first message, etc.). The ZSUBMIT program splits the file into separate messages before submitting them to the server.

Multiple message files

The Zetafax server also supports the sending of multiple message files specified within a SUBMIT file. When a message file is specified in the %%[FILE] section, typically ~SEND000.G3F, .EPN or .TXT in the OUT directory, the Zetafax server will scan for matching files with extension .001, .002 and append these files to the fax message. Additionally, SUBMIT files can also contain more than one %%[TEXT] or %%[FILE] sections, and both ZSUBMIT and the API library functions will generate a series of message files using the new .001 naming convention.

Use with COM and C libraries

SUBMIT files used with the API functions may only contain information for a single message - everything from the first %%[TEXT] command to the end of the file is treated as message text. Certain of the functions only look at one of the two sections - in this case the other section may of course be omitted.

SUBMIT file Message Option lines

These lines appear at the start of the %%[MESSAGE] section, before the addressing lines (i.e. before the first "To:" line). If omitted, all the lines (except the "Comment:", "Attach" and "Charge" lines) default to the values specified for the given user. For files submitted by programs using the API functions, this is the user specified in the ZfxAPIInit call. For files submitted using the ZSUBMIT program, it is the user specified in the User: line, either in the %%[MESSAGE] section (if given), or in the SETUP.INI file.

After

Syntax

```
AFTER: time
```

where time specifies the earliest time that the message should be sent, for deferred sending. The format of the time field may be one of the following:

yy-mm-dd hh:mm:ss (date and time specified)
OFFPEAK(as defined in SETUP.INI)

If the time field is omitted the message is queued for sending immediately.

Description

Specifies when the message is to be sent.

Example

After: 02-07-31 18:00:00

Attach

Syntax

ATTACH: files

where files is a comma separated list of graphics files to attach to the fax.

Description

Specifies the attachment files in the user's private graphics directory (Z-GRAPH) or in the system graphics directory. These are added to the end of the fax before sending. A maximum of 30 files may be specified.

Example

Attach: INFOPACK, PRICES

Charge

Syntax

CHARGE: chargecode

where chargecode specifies the charge code to use for this message, and may be any text.

Description

Specifies the charge code to use for this message. Charge codes are stored in the Zetafax billing log when the message completes, and may be used for charging the message to a particular department or client.

Example

Charge: Sales Dept

Comment

Syntax

COMMENT: comment

where comment is the message description (up to 40 characters long).

Description

Specifies the message description as displayed on the right of the client OUT window. If this line is omitted a message description detailing the first addressee is used.

Example

Comment: Smith and Co January invoice

Coversheet

Syntax

COVERSHEET: coversheet

where coversheet is the name of the file to be used to generate a coversheet for the fax (1 to 8 characters long) or blank for no coversheet.

Description

Specifies what coversheet to generate for the fax onto before sending.

Example

Coversheet: COVSHEET

Covertext

Syntax

COVERTEXT: covertext

where covertext is the next line of covernote text to be added to the coversheet.

Purpose

specifies the text to be added to the coversheet selected in the "COVERSHEET:" message option line. This line can be repeated as many times as required, though all of the COVERTEXT: lines must be together and as they are message options they must occur before the first "To:" line. Zetafax only uses the number of COVERTEXT: lines defined in the coversheet - if the coversheet contains ten "%%[NOTE]" fields, and the submit file has twelve COVERTEXT: lines, the last two COVERTEXT: lines will be ignored.

Example

```
COVERTEXT:Here is the price list as promised.
COVERTEXT: Please call if you have any queries.
COVERTEXT: Regards,
COVERTEXT:Sam Smith
```

Delete

Syntax

DELETE: delete

where delete is either "YES" or "OK".

Purpose

Specifies whether the server should delete a message from the OUT directory after it has completed. If set to "OK" the message will only be deleted if sent successfully, while setting it to "YES" causes it to be deleted on completion whether successful or not.

Example

Delete: YES

Format

Syntax

FORMAT: format

where format specifies the file format of the data file to be sent. The possible values of the format field (together with the data type and the associated data file extensions) are:

ASCII(ASCII text, unformatted - TXT)
EPSON(Epson FX or LQ print spool file - EPN)
TIFF-NORMAL(200x100 dpi TIFF fax file - G3N)
TIFF-FINE(200x200 dpi TIFF fax file - G3F)

Description

Specifies what format the data file is in. In a standard SUBMIT file (containing a %%[TEXT] section) the format should be "EPSON" or "ASCII". Epson should be used if the message text contains any "%%" formatting commands (e.g. "%%[BOLD:ON]"). If however the message text just contains ASCII text then the format should be specified as "ASCII" (or the line omitted), in which case the result will be the same as using the client to send an ASCII text file.

Example

FORMAT: EPSON

From

Syntax

FROM: sendername

where sendername is the originator of the message.

Description

Specifies the name that is put on the fax cover sheet as the sender of the fax (cf the From: field on the Zetafax addressing dialog).

Example

From: Sam Smith

Header

Syntax

HEADER: header

where header is one or more of No, To, Fr, Da and Ti (with no separating spaces), as follows:

No	Page numbering (n/N)
To	Name of recipient
Fr	Name of sender
Dt	Date
Ti	Time

Description

Specifies the format of the header line to appear at the top of each page of the fax.

Example

Header: NoToFrDa

Hold

Syntax

HOLD: hold

where hold is either "YES" or "NO".

Description

Specifies whether the message will be held once it in the queue, meaning that it will not be sent until the user releases it, using the Zetafax client Release command (File Menu). This gives a similar effect to the Hold for preview before sending check box on the Zetafax client addressing dialog, allowing the fax to be checked on screen before sending.

Example

Hold: YES

Letterhead

Syntax

LETTERHEAD: letterhead

where letterhead is the name of the file to be used as the letterhead (1 to 8 characters long) or blank for no letterhead

Description

Specifies what letterhead to merge the fax onto before sending.

Example

Letterhead: LETTHEAD

Preview

Syntax

PREVIEW: preview

where preview is either "YES" or "NO".

Description

Specifies whether a preview file is to be prepared for this message. This is a copy of the fax prepared for the first recipient, which can be viewed using the View (File menu) option on the Zetafax client. The default setting is "YES", but if disk space is limited and there is no requirement to view the faxes from a client then this parameter may be set to "NO".

Note: To preview the fax on the Zetafax client before it is sent (in a similar way to the Hold for preview before sending check box on the Zetafax client addressing dialog) you should set this option to "YES" (the default), and also set the Hold: message option (qv).

Example

Preview: NO

Priority

Syntax

PRIORITY: priority

where priority is one of "URGENT", "NORMAL" or "BACKGROUND".

Description

Specifies the priority to be used when queuing the message, in the same way as the Priority radio button on the client sending options dialog.

Example

Priority: NORMAL

Quality

Syntax

QUALITY: quality

where quality is one of "DRAFT", "NORMAL" or "HIGH".

Description

Specifies the resolution to be used when sending the fax, in the same way as the Resolution button on the client sending options dialog. For fax, "DRAFT" and "HIGH" will normally force the fax to be sent at standard (200x100 dpi) and fine (200x200 dpi) resolutions respectively, whilst "NORMAL" will send at whichever resolution has been specified by the system administrator in the system initialization file.

Example

Quality: NORMAL

Subject

Syntax

SUBJECT: subject

where subject is the message subject (up to 60 characters long).

Description

Specifies the message subject, as displayed on the right of the client OUT window. If a coversheet is being sent, the subject line will be included on the coversheet.

Example

Subject: Smith and Co January invoice

User name

Syntax

USER: username

where username specifies a valid Zetafax user name.

Description

Specifies the Zetafax user submitting this file. The command may be omitted if the default user name specified in the [ZSUBMIT] section in the SETUP.INI initialization file was not blank, or if submitting using the API COM or C language functions.

Note - this command can only be used in files submitted using the ZSUBMIT program (rather than using the API "C" function calls, where the name is specified in the ZfxAPIInit function). The ZSUBMIT program can be configured to disable this command, if desired for permissions or security reasons.

Example

User: JSMITH

SUBMIT file Message Addressing lines

These lines appear in the %%[MESSAGE] section, after the message options lines detailed in the SUBMIT file Message Option lines topic.

Note: A"To:" line must be first line specified (after the last message option line), before any other message addressing lines. The lines may be repeated if the message is to be sent to more than one person, with each repeated set starting with a "To:" line.

Fax
SMS
Field
User name
Organisation
To

Fax

Syntax

FAX: number

where number is the fax telephone number (cf the Zetafax address book editor fax number field).

Description

Specifies the fax number to send the fax to. If the number is unknown at the time of creating the file the number can be omitted. The Zetafax server will reject the send to that recipient, and the message can be resubmitted manually entering the fax number as required.

Default

None - mandatory field (unless LAN: or SMS: line is specified - see below).

Example

Fax: 456 789 0123

SMS

Syntax

```
SMS: number
```

where number is the mobile telephone number of the recipient.

Description

Specifies the mobile telephone number to send the message to. SMS messages are submitted via an SMS center on the mobile network. The telephone number for the message center is configured in the Devices section of the Zetafax Configuration program and it is not necessary to specify it in the SUBMIT file.

Default

None - mandatory field (unless LAN: or FAX: line is specified - see below).

Example

SMS: 456 789 0123

Field

Syntax

FIELD: fieldname text

where fieldname is a single word (i.e. without any spaces) giving the name of a Embedded Command field, and text is the string which will be substituted for the field.

Description

Embedded Command fields are identified in a document by the string "`%%[fieldname]`" (without the double quotes). When the fax is prepared for sending these fields are replaced with the text given, allowing faxes to multiple addressees to be personalized, for example. Multiple FIELD lines may be specified.

Note: The Field lines are not retained if a message is resubmitted using the Zetafax client program (for example, to correct a fax number).

The following field names are reserved, and should not be used with this command:

- COVERTEXT
- ORGANISATION
- DATE
- PAGE
- FONT
- PAGES
- FROM
- PORTRAIT
- LANDSCAPE
- SUBJECT
- NAME
- TIME
- NOTES

Default

None - any fields not specified are treated as blank.

Example

Field: Department Sales and Marketing

User name

Syntax

LAN: username

where username is the Zetafax username of the person to be sent to.

Description

Specifies the name of the Zetafax user for messages which are to be sent across the LAN (appearing in the user's IN window). This can be useful when testing applications to check the appearance of faxes to be sent without actually sending them to the remote user, in the same way as one might use Preview from the client.

Default

None - used as an alternative to the Fax: line.

Example

LAN: Fsmith

Organisation

Syntax

ORGANISATION: organisation

where organisation is the name of the company or organisation to which the fax recipient belongs.

Description

The organisation name is used on the coversheet of fax messages.

Default

Blank.

Example

Organisation: Smith and Sons Limited

To

Syntax

TO: name

where name is the name of the person to whom the message is to be sent.

Description

Specifies a recipient of the fax (cf the Coversheet Name field in the Zetafax address book). This name is used on the coversheet of the fax, on the header line of the fax, and also in the comment on the right hand side of the OUT window in the client display, Unless overridden by the comment message options line.

Default

None - mandatory line.

Example

To: Sam Smith

SUBMIT file Message Text commands

These commands appear in the %%[TEXT] section and may be freely interspersed with the message text.

See the Embedded Command description in SUBMIT file Message Addressing lines for details about what to do if you need to include the characters "%%[" in the message, without them being treated as a command.

Append

Syntax

%%[APPEND:file]

where *file* is the base name of a graphics file in the Z-GRAPH directory.

Description

Appends a graphics image (e.g. a scanned information sheet) to the end of the message (similar to the Zetafax client **Attach** option). Up to 30 of these commands may be entered, and the named files will be added to the end of the message as new pages. This command can also be used in separate ASCII format files specified using the %%[FILE] option.

Example

%%[Append:INFOPAGE]

Bold

Syntax

%%[BOLD:state]

where *state* is either "ON" or "OFF".

Description

The %%[BOLD:ON] command causes all characters following to be printed in bold within the fax message until either a %%[BOLD:OFF] command is encountered or the end of the %%[TEXT] section is reached.

Example

%%[Bold:ON]

Date

Syntax

%%[DATE]

Description

Inserts the date when the message is prepared for sending, in the format "31st August 2005". This command can also be used in separate ASCII format files specified using the %%[FILE] option.

Example

%%[Date]

Down

Syntax

%%[DOWN:distance]

where *distance* is the number of units to move down the page from the current position. A unit is approximately 1/100th of an inch.

Description

This command moves down the page a given distance and may be used, for example, to position the top line of text to fit within a merged form.

Note: The first line of text prints approximately 0.7 inches below the top of the page.

Example

%%[Down:100]

Field name

Syntax

%%[fieldname]
where fieldname is the name of an Embedded Command.

Description

Inserts the text for the given field for that addressee. The following fieldnames are defined for all messages submitted using the API or ZSUBMIT program:

- FROM
- NAME
- ORGANISATION

Other fields are only defined if they have been specified using the Field: message addressing line (see SUBMIT file Message Addressing lines). This command can also be used in separate ASCII format files specified using the %%[FILE] option.

Example

```
%%[Organisation]
```

Font

Syntax

%%[FONT:font{,size}{,B}{,I}]

where *font* is the font required, and is any TrueType font available on the Zetafax server. Alternatively, for compatibility with previous versions, one of "ROMAN10", "ROMAN12", "ROMAN20", "ROMAN5", "ROMAN6" or "ROMANPS" gives 10cpi, 12cpi, 20cpi, 5cpi and 6cpi fixed space fonts, and proportional spaced font respectively.

Description

This command is used to set the font to be used for the remainder of that message (i.e. until the end of the %%[TEXT] section).

Example

%%[Font:Arial,10]

Insert

Syntax

%%[INSERT:file]

where *file* is the base name of a graphics file in the Z-GRAPH directory.

Description

Inserts a single page graphics image (e.g. a signature) at the current position on the page. The image is merged into the faxed document - you must leave enough lines following the command blank if you wish to avoid the image overprinting the following text. This command can also be used in separate ASCII format files specified using the %%[FILE] option.

Example

%%[Insert:ABCLOGO]

Landscape

Syntax

%%[LANDSCAPE]

Description

This command causes the text which follows (until a %%[PORTRAIT] command or the end of this message, whichever comes first) to be printed in landscape orientation. The command must come before any other text - i.e. at the start of the first line following the %%[MESSAGE] command, or immediately after a %%[PAGE] command (on the same line). This command can also be used in separate ASCII format files specified using the %%[FILE] option.

Example

%%[Landscape]

Left margin

Syntax

%%[LMARGIN:width]

where *width* is the size of the left hand margin in character widths. The width of each character is dependent upon the font being used.

Description

This command is used to set the left margin for the remainder of that message (i.e. until the end of the %%[TEXT] section). It should be placed at the start of a line.

Note: The margin width excludes a strip of width 0.5 inches at the left of the page which may not be used (i.e. a value of 0 would set the left hand margin 0.5 inches from the left of the page).

Example

%%[LMargin:200]

Page

Syntax

%%[PAGE]

Description

This command is used to start a new page.

Example

%%[Page]

Portrait

Syntax

%%[PORTRAIT]

Description

This command cancels a previous %%[LANDSCAPE] command, reverting to portrait orientation. The command must come before any other text - i.e. at the start of the first line following the %%[MESSAGE] command, or immediately after a %%[PAGE] command (on the same line).

Note: The margin settings following a %%[PORTRAIT] command are different from the defaults for ASCII files, allowing slightly more of the page to be printed. This can be useful when overprinting forms, for example. This command can also be used in separate ASCII format files specified using the %%[FILE] option.

Example

%%[Portrait]

Right

Syntax

%%[RIGHT:distance]

where *distance* is the number of units to the right from the current position. A unit is approximately 1/100th of an inch.

Description

This command moves across the page to the right a given distance and may be used, for example, to position a word precisely within a form.

Example

%%[Right:100]

Time

Syntax

%%[TIME]

Description

Inserts the time when the message is prepared for sending, using a 24 hour clock, in the format "12:34". This command can also be used in separate ASCII format files specified using the %%[FILE] option.

Example

%%[Time]

Underline

Syntax

%%[UNDERLINE:state]

where *state* is either "ON" or "OFF".

Description

The %%[UNDERLINE:ON] command causes all characters following to be printed underlined until either a %%[UNDERLINE:OFF] command is encountered or the end of the %%[TEXT] section is reached.

Example

%%[Underline:ON]

Action commands

These remaining commands may be used within a document to split it into a number of faxes.

Send
Preview

Send

Syntax

%%[Send]

Description

Indicates that the document being printed should be split at the foot of this page and submitted as a fax. The remaining page or pages will be treated as a separate fax or faxes.

Example

%%[Send]

Preview

Syntax

%%[Preview]

Description

Indicates that the document being printed should be split at the foot of this page and submitted as a fax for preview. The remaining page or pages will be treated as a separate fax or faxes.

Example

%%[Preview]

Example fax SUBMIT files

Simple message

```
%%[MESSAGE]
User: JJONES
From: Jim Jones
To: Sam Smith
Organisation: Smith and Sons
Fax: 456 789 0123

%%[TEXT]
Dear Sam Here's a short fax produced automatically Yours, Jim
```

Multiple messages

```
%%[MESSAGE]
User: JJONES
From: Jim Jones
To: Sam Smith
Organisation: Smith and Sons
Fax: 456 789 0123
%%[TEXT]
Hello Sam!

%%[MESSAGE]
User: JJONES
From: Jim Jones
Coversheet: COVSHEET
To: Alan Ayton
Fax: 456 789 0123
%%[TEXT]
Hello Alan!
```

Multiple addressees

```
%%[MESSAGE]
User: SSMITH
From: Sam Smith
Coversheet: COVSHEET
Letterhead: LETTHEAD
To: Sam Smith
Organisation: Smith and Sons
Fax: 456 789 0123

To: Jim Jones
Organisation: Jones Brothers
Fax: 123 456 7890
%%[TEXT]
Don't forget the steering group meeting this evening Sam
```

Multiple message files

```
%%[MESSAGE]
User: JJONES
From: Jim Jones
To: Sam Smith
```

Organisation: Smith and Sons
Fax: 456 789 0123
%%[TEXT]
Here are the brochure and price list I promised to send.I've included the general brochure as I thought this would be more helpful.
%%[FILE]
C:\Program Files\Zetafax Server\SERVER\Z-TEMP\BROCHURE.G3F
%%[FILE]
C:\Program Files\Zetafax Server\SERVER\Z-TEMP\PRICES.G3F
%%[TEXT]
Please call me if you have any other questions
Text formatting

%%[MESSAGE]
User: JJONES
From: Jim Jones
Coversheet: COVSHEET
Letterhead: LETTHEAD
Syntax EPSON
To: Sam Smith
Organisation: Smith and Sons
Fax: 456 789 0123

%%[TEXT]
%%[LMARGIN:200]
%%[DOWN:200]
%%[FONT:ROMAN10]

Dear Jim Here's a%%[BOLD:ON]short fax%%[BOLD:OFF] produced automatically %%[PAGE]%%[DOWN:200]This is the second page Now to end the letter with a signature added using the standard "insert graphics" command. Yours%%[INSERT:JIMSIG] Jim Jones

Sending SMS messages

The Zetafax server is capable of submitting Short Message Service (SMS) or text messages to mobile devices using the commands specified below. SMS messages are submitted via an SMS center before being delivered to the mobile device.

SMS messages are limited to 160 characters per short message. Longer messages can be sent using Zetafax, the Zetafax server handles longer SMS messages by breaking one long message into several smaller messages. Preferences for handling longer messages can be modified in the **Zetafax Configuration** program under the **Devices** note for each SMS device you have configured.

Message commands

SMS messages can handle message text only, and cannot be used to send graphics files or other file formats. As a result, several Message Option and Message Text commands that can be used to create fax messages are not relevant when sending SMS messages. It is also important to include "Coversheet: ". This denotes that you do not wish to use a coversheet which are not supported by SMS meessages.

The following is a list of Message Option commands that are supported when sending SMS messages. For a detailed description of each command, refer to SUBMIT file Message Option lines.

Command	Description
After	Specifies the earliest time that the message should be sent.
Charge	Specifies a charge code to use for a message.
Comment	Specifies the message description.
Delete	Specifies whether the Zetafax server should delete the message from the OUT directory after it has completed.
Format	Specifies what format the data file is in.
Hold	Specifies whether the message will be held once it in the queue, meaning that it will not be sent until the user releases it .
Priority	Specifies the priority to be used when queuing the message.
Subject	Specifies the message subject, as displayed in the client OUT window.
User	Specifies the Zetafax user submitting this file.

The following Message Option commands are not applicable and are therefore not supported when sending SMS messages using ZSUBMIT:

- ATTACH,
- COVERSHEET,
- COVERTEXT,
- FAX,
- FIELD,
- FROM,
- HEADER,
- LAN,
- LETTERHEAD,
- ORGANISATION,
- PREVIEW,
- QUALITY.

The following Message Text commands are not supported when submitting SMS messages using ZSUBMIT:

- %%[APPEND],
- %%[BOLD],
- %%[DOWN],
- %%[FONT],
- %%[INSERT],

- %%[LANDSCAPE],
- %%[LMARGIN],
- %%[PAGE],
- %%[PORTRAIT],
- %%[RIGHT],
- %%[UNDERLINE].

Example SMS SUBMIT files

Simple message

```
%%[MESSAGE]
User: JJONES
Coversheet:
To: Sam Smith
SMS: 456 789 0123

%%[TEXT]
Dear Sam
Here's a short SMS message produced automatically
Yours, Jim
```

Multiple messages

```
%%[MESSAGE]
User: JJONES
Coversheet:
To: Sam Smith
SMS: 456 789 0123

%%[TEXT]
Hello Sam!

%%[MESSAGE]
User: JJONES
Coversheet:
To: Alan Ayton
SMS: 123 456 7890

%%[TEXT]
Hello Alan!
```

Multiple addressees

```
%%[MESSAGE]
User: JJONES
Coversheet:
To: Sam Smith
SMS: 456 789 0123
Coversheet:
To: Alan Ayton
SMS: 123 456 7890

%%[TEXT]
Don't forget the steering group meeting this evening Sam
```

Multiple message files

```
%%[MESSAGE]
User: JJONES
Coversheet:
To: Sam Smith
SMS: 456 789 0123
```

%%[TEXT]
Here's the price list I promised to send to you.

%%[FILE]
C:\Program Files\Zetafax Server\SERVER\Z-TEMP\PRICES.TXT
%%[FILE]
C:\Program Files\Zetafax Server\SERVER\Z-TEMP\BUSCARD.TXT

%%[TEXT]
Please call me if you have any questions.

Submitting SUBMIT files

When using the ZSUBMIT program, files in SUBMIT file format may be submitted by simply copying them to a file in the specified directory with the extension ".SUB". If the file contains more than one "%% [MESSAGE]" section (i.e. contains more than one message to be sent) then it is first split up by the program into separate message files in the polling directory, named ~ZSUBnnn.SUB, where "nnn" is a unique number for each file. Each file is then interpreted separately and submitted to the server for sending.

Syntax errors

If an error is found in the format of one of the files (e.g. an invalid or missing line in the message options), an error message is logged giving the problem and file name, and the file extension is changed to ".ERR"; otherwise the file is deleted once it has been separated (in the case of multiple message files) or submitted successfully. This directory should be checked if submit errors occur to identify the problem then delete the ".ERR" file - performance of the program will be affected if the number of files in the directory is allowed to become very large.

Automatic deletion

ZSUBMIT deletes SUBMIT files automatically once they have been processed and submitted to the Zetafax server. It will also delete matching files with extensions .001, .002 etc., continuing until it does not find the next file in sequence. This makes it simple to submit messages comprising multiple files, without having to tidy up manually.

Conversely, if a file is used by more than one SUBMIT file, it should not be given an extension of .001.

File creation

It is important to ensure that the submit file is complete (and any files it references exist), from the moment when the ZSUBMIT program might try and read it.

The ZSUBMIT program will behave correctly if the .SUB file is locked, retrying until it can access the file; however this can still give inaccurate results with some applications. With some programs, the file can be accessed and is "readable" even while it is being created, whilst others can create a zero length file first before reopening it to write the data. The symptom is occasional failures when ZSUBMIT tries to read the file at the precise moment it is being created.

The best and recommended approach is to create the file with a different extension e.g. .NEW and rename it to .SUB only when it is complete.

Server settings options for sending ASCII text files as messages

You can set up Zetafax to send ASCII text files as messages in the **Server settings** options of the Zetafax Configuration program. These options are used if a text file is sent directly using the Zetafax Client **File Send** option, rather than printed from a word processor (using the Zetafax printer driver).

- To display this dialog, double-click **Sending ASCII text files** in **Server settings**, or select the **Text files** icon from the **Zetafax - configuration options** dialog.

Top margin

Range 0 - 15 lines.

Specifies the number of blank text lines at the top of the fax page when preparing an ASCII text file for sending.

Bottom margin

Range 0 - 15 lines.

Specifies the number of blank text lines at the bottom of the fax page when preparing an ASCII text file for sending.

Page length

Range 35 - 100 lines.

Specifies the number of text lines per page (including the top and bottom margins) when preparing an ASCII text file for sending. A value of 72 gives approximately an A4 page.

OK

Save any changes that have been made to the ASCII text files configuration, and exit the Zetafax - configuration options dialog.

Cancel

Do not save changes made to the ASCII text files configuration, and exit the Zetafax - configuration options dialog.

Reset

Reset the ASCII text files configuration to its default settings. The settings are not saved until either **OK** is clicked or another category icon is selected from the scroll (left).
If you use the scrolling icons on the left to move to another **Category**, any changes made to the ASCII text files configuration are saved.

Index

- A -

- C -

- D -

- E -

- F -

- G -

- I -

- S -

- Z -