Genifax COM API
User Guide

March 2003

Omtool Headquarters
8 Industrial Way
Salem, New Hampshire 03079
Phone (603) 898-8900
Toll Free (888) 466-8665
Fax (603) 890-6756

Omtool Europe
Cobb House
2-4 Oyster Lane
Byfleet, Surrey KT14 7DU
United Kingdom
Phone +44 1932 33 4444
Fax +44 1932 34 2316

rev 4.0269-AC March 2003

CONTENTS

Chapter 4 - Properties and Methods

Preface

What you will find in this section...

About This Manual

The Genifax COM API User's Guide provides pertinent information to programmers adding Genifax functions to their applications. It describes how to install the COM API, provides step-by-step guidelines for coding basic functions such as sending a message and retrieving status, and includes details about the properties and methods used by the COM API.

Assumptions

This book is written for experienced applications developers who are familiar with Microsoft Windows programming conventions and the Visual Basic language. Familiarity with the functions available in the Omtool Genifax product is required.

Conventions

This document uses the following formatting conventions:

Bold	Highlights the names of properties and methods. Get the **Messages** property of the MessageServer object. Used to emphasize controls and buttons. Click **Connect**.
Italic	Identifies arguments used by a method. object.**ConnectToServer** (*servername, username*)

`Courier font`	Used for programming examples, code snippets, and file names.
	`' Create a new ServerConnect object`
	`Set con = New ServConnect`
	`Set server = con.ConnectToServer _`
	`(ServerName, LoginID)`

Single quote (')	Precedes comments within the code. Comments are ignored when the code is processed.
Underscore (_)	A continuation character that indicates the line of code is too long to fit on a single line in the book, but is a single line in the program. The underscore is *not* part of the code.

Organization

The information contained in this document is presented in four chapters:

Chapter 1 — Introduces the Omtool Genifax COM API, describes what you can do with it, and lists the components that comprise the product.

Chapter 2 — Describes how to install the COM API and how to run the sample application to verify that everything is working correctly.

Chapter 3 — Explains some basic concepts, provides an overview for coding send and administrative functions, and includes programming tips.

Chapter 4 — Alphabetically lists and describes the COM API properties and methods.

Related Documentation

For information about the Omtool Genifax product, refer to the online documentation that is shipped with those products.

Contacting Omtool

To contact Technical Support:

- Call (888) 303-8098
- Send an e-mail to support@omtool.com
- Send a fax to (603) 898-3592
- Visit the Support web site:
  http://www.omtool.com/support/

To contact Customer Service:

- Call (888) 303-8098
- Send an e-mail to customerservice@omtool.com
- Send a fax to (603) 890-1948
- Complete the online form on the Omtool web site:
  http://www.omtool.com/contact/service.cfm

To contact Omtool Europe:

- Call +44 1932 33 4444
- Send an e-mail to euro.support@omtool.com
- Send a fax to +44 1932 34 2316

1 Introduction

What you will find in this chapter...

What is the Omtool Genifax COM API?

The Genifax COM API (hereafter referred to as the COM API) is a software development tool that enables you to add fax and secure messaging functions to existing applications or to create new applications based on these capabilities. The functions available are a subset of Omtool's popular Genifax product.

Genifax enables you to send a wide variety of files — including Microsoft Word and Excel files plus image files such as BMP, GIF, and JPG — as fax messages.

The COM API is intended for experienced applications developers. You should be familiar with Microsoft Windows programming conventions and know how to code with Microsoft Visual Basic version 6.0 or later. You should also be familiar with the functions available in the Omtool Genifax product.

The COM API should be installed in a Microsoft Windows 2000 development environment that includes Microsoft Visual Basic version 6.0 or greater. Applications created with the COM API can run on any industry-standard personal computer (PC) platform with a Microsoft Windows NT or Windows 2000 operating system, and access to a computer running the appropriate Omtool messaging software.

Features

The COM API includes 32-bit dynamic link libraries (DLLs), which expose the API as Visual Basic objects, properties, and methods. It enables you to quickly add basic fax capabilities to an existing application by setting a few properties and calling several methods.

The Genifax COM API supports the following functions:

- Create and send a fax message to one or more recipients
- Specify the time after which the fax will be sent
- Select a template for a cover page and populate the fields with appropriate information
- Retrieve fax and recipient status after the fax is sent

Components

The Genifax COM API software is distributed on a CD-ROM that contains:

- Libraries and support files required to develop applications
- Sample application
- Genifax COM API documentation in Adobe Acrobat® PDF format

2 Software Installation

What you will find in this chapter...

Installing the Software

This chapter describes how to install the Genifax COM API software and how to run the sample application to confirm that everything is working properly.

Before You Begin

Before you install the COM API software, ensure that your computer meets the following criteria:

- Personal computer (PC) with sufficient speed, memory, and storage to support application development and testing
- Microsoft Windows 2000 (Professional or Server) operating system with Service Pack 1 or later
- Microsoft Visual Basic version 6.0 or greater
- Access to an Omtool Genifax server

Additionally, you must have a license from Omtool that enables the installation and use of the Genifax COM API. See *Contacting Omtool* on page 8, for information about contacting Omtool if you do not already have a license number for the Genifax COM API.

Installing the COM API

Perform the following steps to install the COM API software:

1. Download the GenifaxComAPI.exe file from the Omtool Web Center through the Genifax Server Administrator.

   Note: If you do not have access to the Genifax Server Administrator, contact the administrator for assistance with downloading the COM API installation program.

2. In the folder into which you downloaded the installation program, double-click on GenifaxComAPI.exe.
3. Click **Next** on the Welcome dialog box. The Omtool License Agreement appears.
4. Read the agreement, and if you agree to the terms, click the appropriate option and click **Next**.
5. Type your user name, the name of your company, and the license number you received from Omtool in the appropriate text boxes on the Customer Information dialog box, and then click **Next**.

6. Select an installation option on the Setup Type dialog box and click **Next**.

   Note: A complete installation is recommended.

7. Do one of the following based on your selection on the Setup Type dialog box:

   - If you selected Complete, proceed to step 8.
   - If you selected Custom, the Custom Setup dialog box appears. Select the features you want to install and specify where you want to install them and click **Next**.

8. On the Ready to Install the Program dialog box, click **Install**. After the appropriate files have been copied to your system, the InstallShield Wizard Completed dialog box appears.
9. Click **Finish**.
10. Restart your computer so that changes made by the installation program will take effect.

Testing the Configuration

Before you create an application with the COM API, you should confirm that your computer is connected to a Genifax Server. Later, if you have problems getting your application to work, you can rule out an invalid configuration as the cause.

You can test your configuration by using the sample application to send a message. Perform the following steps:

1. Start Visual Basic.
2. Locate and open the sample application's project file:

   `SimpleSample.vbp`

   If you installed the COM API software in the default location, look for the project file in the following folder:

   `Program Files\Omtool\COM API\Sample`
3. Open the module MainSample (main.bas).
4. Verify the following conditions, and modify the settings in the MainSample module if needed:
   - If the Genifax Server is remote from the computer on which you installed the COM API, set ServerName to the name of your Genifax Server. For example: `servername = GFServer`
   - If the COM API was not installed into the default location, set InstallFolder to the location into which you installed the API. For example: `InstallFolder = c:\OmtoolComApi`
   - If the Genifax Server is remote from the computer on which the COM API is installed, make sure that the path to the location of the COM API in InstallFolder is formatted as a UNC path. For example:
     `InstallFolder = \\Program Files\Omtool\COM API\`
5. Set RecipientMessageType to the following value: `Fax`

   For example: ***`RecipientMessageType = Fax`
6. From the Visual Basic Run menu, select **Start**.

   The program sends a test message to the Genifax Server and creates a file in the COM API installation directory called output.txt that contains the results of the test message.

3 Programming Overview

What you will find in this chapter...

Classes

The following classes provide the properties and methods used by Genifax functions:

Class	***Description***
Container	Holds message and recipient data.
Message	Provides information about the message.
MessageAttachment	Provides information about attachments.
MessageRecipient	Provides information about recipients.
MessageServer	Supports the **Messages** and **Recipients** properties.
ObjectCollection	Holds a collection of objects — either attachments or recipients.
ServConnect	Establishes a connection with the Genifax Server.

Concepts

This section describes some concepts that you should understand before programming with the COM API.

Message

A message is an object consisting of one or more *attachments* and one or more *recipients*. To be valid, a message must contain at least one attachment and one recipient.

Genifax messages are converted by the Genifax Server into TIF files.

To build a message, you create a Message object, set the properties, and then add attachments and recipients.

Attachment

MessageAttachment objects are electronic data — such as documents, spreadsheet files, cover pages, and image files — that become part of the message. Attachments are converted into either Genifax messages by the Omtool Genifax Server.

Recipient

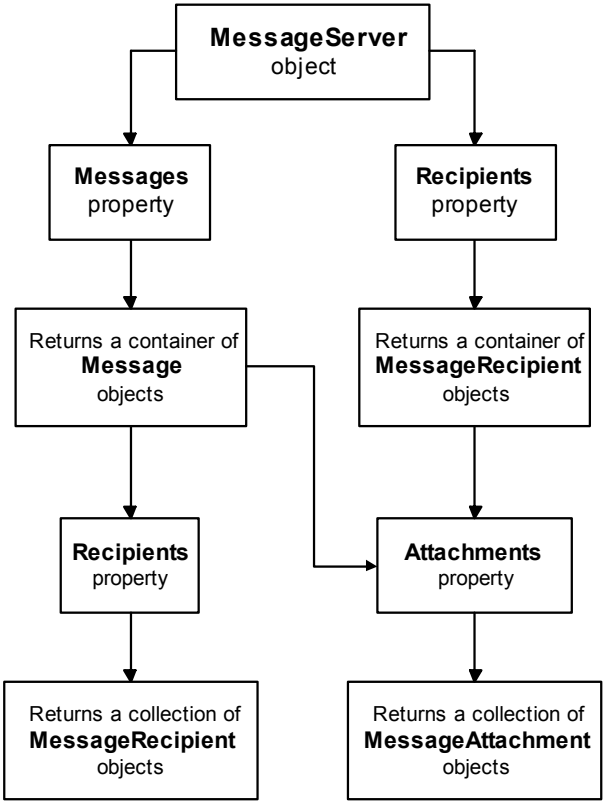
A MessageRecipient is an object that contains information about where the message attachments are being sent. To be valid, a recipient must have a destination (e-mail address or telephone number) set. All recipients of a message must be of the same type.

User

A User is an object that contains information about Genifax users.

Relationship Between Objects and Properties

The following illustration shows the relationship between several objects and properties.



Sending Messages

The procedures for sending a Genifax are described in the following sections. To view sample code that you can use to perform these procedures, open the sample project, `SimpleSample.vbp`.

Sending a Genifax Message

The following steps describe how to connect to the Genifax Server, create a new fax message, and then send it:

1. Create a new ServConnect object by calling the **ConnectToServer** method and returning the results to a MessageServer object.
2. Create a new Container object.
3. Get the **Messages** property of the MessageServer object and return the results to the new Container object.
4. Create a new Message object.
5. Set the following Message object properties if you want — all are optional.
   - **BillingCode**
   - **Priority**
   - **TemplateFileName**
6. For each attachment you want to add, perform steps a through c.
   a. Create a new MessageAttachment object.
   b. Set the **AttachmentOriginalPathname** property of the MessageAttachment object to the file name and location of the attachment.
   c. Add the attachment to the message by calling the **Add** method of the Message object.
7. For each recipient you want to create, perform steps a through c.
   a. Create a new MessageRecipient object.
   b. Set the following MessageRecipient properties — Destination is required, the others are optional.
      – **BillingCode**
      – **DateSendAfter**
      – **Destination** (required)

– **RecipientType** (defaults to fax)

– **PreviewRequired**

– **Priority**

– **TemplateFileName**

c. Add the recipient to the message by calling the **Add** method of the Message object.

8. Deliver the message to the message server by calling the **Save** method of the Message object.

After you have sent all of your faxes, refer to the section "Message Administration" for details about retrieving status information.

Message Administration

The procedures for retrieving the status of a Genifax message are similar, and are described in the following sections. To view sample code that you can use to perform these procedures, open the sample project, `SimpleSample.vbp`.

Retrieving Status for a Genifax Message

The following steps describe how to connect to the Genifax Server and retrieve status of Genifax messages:

1. Create a new ServConnect object by calling one of the following methods and returning the results to a MessageServer object:
   - **ConnectToServer** method to view status for faxes sent by the user who is logged on.
   - **ConnectToServerAdmin** method to view status for all faxes sent.
2. Each time you want to update status, perform steps a through c.
   a. Get the **Messages** property of the MessageServer object and return the results to a Container object to display status for each message.
   b. Get the **Recipients** property of the MessageServer object and return the results to a Container object to display status for each recipient.
   c. Retrieve the status for additional fax properties by looping through each Message or MessageRecipient object in the container and getting the desired properties for each one.

Additional Information

Handling Errors

Good programming practice includes identifying and fixing run time errors whenever possible, providing information to the user to help correct the problem, or as a last resort, shutting the program down gracefully. You can use the On Error GoTo command to transfer program control to routines that perform these actions.

You can also use the Select Case statement to step through different possibilities, selecting an action to perform when predetermined criteria have been met.

For more information about handling errors, and for help in identifying run-time errors, refer to the Microsoft Visual Basic help.

Distributing Visual Basic Applications

The Package and Deployment Wizard included with Visual Basic is a convenient way to identify and package the supporting files you will need to ship with your application. Applications you create with the COM API will typically require the following support files:

- Msvbvm60.dll — a Microsoft file that is required when you run any program created with Visual Basic 6
- Other required files — any additional files to support custom controls that your application uses
- Omtool support files — see the following section of this document

Refer to the Microsoft Visual Basic documentation for more information about support files and for running the Package and Deployment Wizard.

Omtool Support Files

The program that installs your application must install several Omtool support files. If these files are not installed, unexpected behavior within your program can result.

File	***Comments***
`omfscriptingu.dll`	***The COM API — must be registered by the installation.
`oming.dll`	***A support file — must be installed in the same folder as `omfscriptingu.dll`.

File	*Comments*
`omfInterfacesps.dll`	***The proxy stub — must be installed with the Share attribute enabled, and registered by the installation program. Because this file is shared by other Genifax Server applications, install it in the following location: `Drive:\Program Files\Common Files\Omtool`

4 Properties and Methods

What you will find in this chapter...

Properties

This section lists the Genifax COM API properties alphabetically and provides information about each one. In some cases, code snippets from the sample application are included in the Examples section of a property.

AccessCode Property

Description: Returns or sets the sender's access code.

Applies To: **Message** object

Usage: object.**AccessCode** [= *value*]

Data Type: String

Remarks: Access codes are additional dialing codes that some telephone systems require in order to place a call.

ApprovalDelegates Property

Description: Returns a list of the users who can approve this user's sent faxes.

Applies To: **User** object

Usage: set [collection object] = object.**ApprovalDelegatesProperty**

Data Type: Object collection

Remarks: This property is read-only.

ApprovalSubordinates Property

Description: Returns a list of the users whose faxes this user can approve.

Applies To: **User** object

Usage: set [collection object] = object.**ApprovalSubordinatesProperty**

Data Type: Object collection

Remarks: This property is read-only.

AttachmentOriginalPathName Property

Description: Returns or sets the original path and file name of the message attachment.

Applies To: **MessageAttachment** object

Usage: object.**AttachmentOriginalPathName** [= *value*]

Data Type: String

Remarks: When setting this property, use a fully-qualified file name that includes a path to the file location.

Example:

```
Dim MessageBody As New MessageAttachment
frmBuildFax.rtbMessageBody.SaveFile _ "MessageBody.rtf"
MessageBody.AttachmentOriginalPathName =
_ "messageBody.rtf"
APISampleMessage.Add MessageBody
```

AttachmentPageCount Property

Description: Returns the size of the attachment file in bytes.

Applies To: **MessageAttachment** object

Usage: object.**AttachmentPageCount**

Data Type: Long

Remarks: This property is read-only, and pertains to Release 2.4.2 or later.

AttachmentSize Property

Description: Returns the number of pages in the attachment.

Applies To: **MessageAttachment** object

Usage: object.**AttachmentSize**

Data Type: Long

Remarks: This property is read-only, and pertains to Release 2.4.2 or later.

Attachments Property

For the Message object:

Description: Returns a collection of all the MessageAttachment objects in the message.

Usage: set [collection object] = object.**Attachments**

Data Type: Object collection

Remarks: This property is read-only.

For the MessageRecipient object:

Description: Returns a collection of all the MessageAttachment objects for the recipient.

Usage: set [collection object] = object.**Attachments**

Data Type: Object collection

Remarks: This property is read-only.

BillingCode Property

For the Message object:

Description: Returns or sets a value for the billing code.

Usage: object.**BillingCode** [= *value*]

Data Type: String

Remarks: Billing codes are used by some companies to track faxing activity.

Maximum string length is 50 characters.

For the MessageRecipient object:

Description: Returns or sets a value for the billing code.

Usage: object.**BillingCode** [= *value*]

Data Type: String

Remarks: Billing codes are used by some companies to track faxing activity.

Setting this property will override the **BillingCode** property value set from the Message object.

Maximum string length is 50 characters.

Example:
'Set BillingCode, Priority,
TemplateFileNameAPISampleMessage.Priority =
cmbMessagePriorityAPISampleMessage.BillingCode =
_ txtMessageBillingCodeAPISampleMessage.TemplateFilena
me = _ txtMessageCoverPage

CanApproveFaxes Property

Description: Returns a value to indicate whether the Genifax Web Client should display the Approval tab for the user.

Applies To: **User** object

Usage: object.**CanApproveFaxes** [= *value*]

Data Type: Boolean

Remarks: This property determines whether a user can approve the outbound faxes of other Genifax users. The value of this property can be true or false. When the value is true, the user can approve the outbound faxes of other Genifax users.

CancelPending Property

Description: Returns a value to indicate that the sender requested the message be cancelled.

Applies To: **MessageRecipient** object

Usage: object.**CancelPending**

Data Type: Long

Completed Property

Description: Returns an indication that the recipient is in its final state and ready to be sent.

Applies To: **MessageRecipient** object

Usage: object.**Completed**

Data Type: Long

Remarks: This property is read-only.

Count Property

For the Container object:

Description: Returns the number of objects in the container.

Usage: object.**Count**

Data Type: Integer

Remarks: This property is read-only.

For the ObjectCollection object:

Description: Returns the number of elements in the collection.

Usage: object.**Count**

Data Type: Integer

Remarks: This property is read-only.

DateCompleted Property

For the Message object:

Description: Returns the date and time at which the message was fully completed.

Usage: object.**DateCompleted**

Data Type: Date

Remarks: "Completed" means the processing of all of a message's recipients has been completed.

This property is read-only.

For the MessageRecipient object:

Description: Returns the date and time at which the recipient reached its final state.

Usage: object.**DateCompleted**

Data Type: Date

Remarks: The final state is when processing of the recipient is complete.

This property is read-only.

DateSendAfter Property

Description: Returns or sets a date and time after which the recipient's attachments will be sent.

Applies To: **MessageRecipient** object

Usage: object.**DateSendAfter** [= *value*]

Data Type: Date

Remarks: This property is used to delay sending a recipient's attachments to the Genifax Server.

DateSubmitted Property

For the MessageRecipient object:

Description: Returns the date and time at which the message was submitted to the message server.

Usage: object.**DateSubmitted**

Data Type: Date

Remarks: This property is read-only.

For the MessageRecipient object:

Description: Returns the date and time at which the recipient was submitted to the message server.

Usage: object.**DateSubmitted**

Data Type: Date

Remarks: This property is read-only.

Destination Property

Description: Returns or sets the address of a message recipient.

Applies To: **MessageRecipient** object

Usage: object.**Destination** [= *value*]

Data Type: String

Remarks: This value is typically a fax telephone number or an e-mail address.
(See example on next page.)

Example:
'Set RECIPIENT Destination, Template file name, 'and Billing codeFaxRecipientName.Destination = frmMessage _ Check.lstMsgCheckAddressees.List _ (ThisRecipient - 1)FaxRecipientName.TemplateFilename = _ frmMessageCheck.txtFirstCoverPageFaxRecipientName.BillingCode = _ frmMessageCheck.txtFirstBilling

DestinationLocalized Property

Description: Returns the final destination of the message.

Applies To: **MessageRecipient** object

Usage: object.**DestinationLocalized**

Data Type: String

Remarks: The final destination for a message is either a fax number or a destination specified by a routing rule.

This property is read-only.

EnableFaxManagerCompose Property

Description: Returns whether the Genifax user can compose faxes using the Genifax Web Client.

Applies To: **User** object

Usage: object.**EnableFaxManagerComposer** [= *value*]

Data Type: Boolean

Remarks: The value for this property can be true or false. When the value is true, the Genifax user can create and send faxes using the Genifax Web Client.

FaxCenterGenerated Property

Description: Returns whether an inbound message was delivered to a user from FaxCenter.

Applies To: **MessageRecipient** object

Usage: object.**FaxCenterGenerated**

Data Type: Boolean

Remarks: When this value is true, the message was routed from FaxCenter.

FaxManagerDelegates Property

Description: Returns a list of the users who can view this user's fax information.

Applies To: **User** object

Usage: object.**FaxManagerDelegates** [= *value*]

Data Type: Object collection

Remarks: Creates, modifies, and deletes delegates. (See example on next page.)

Example:
```
Dim Delegate As UserProxyfor each user in Server.Users  'create a new delegate and set permissions   set delegate =  user.FaxManagerDelegates _
.New  delegate.Email=
"fjones@company.com"delegate.ViewFaxes = true
  delegate.SendOnBehalf = false
  delegate.Save
  'delete each delegate for each delegate in user.FaxManager _
  Delegates
      delegate.delete
  next
next
```

FaxManagerSubordinates Property

Description: Defines whose faxes the specified user can view and whether the specified user can send faxes on behalf of the other user.

Applies To: **User** object

Usage: object.**FaxManagerSubordinates** [= *value*]

Data Type: Object collection

FinalFormCode Property

Description: Returns the file format in which the message is being sent to the recipient.

Applies To: **MessageRecipient** object

Usage: object.**FinalFormCode**

Data Type: String

Remarks: The final form is usually a TIFF file.

This property is read-only.

Findwhere Property

Description: Returns messages that meet the criteria you specify in a Boolean expression.

Applies To: **MessageServer** object

Usage: object.**FindWhere**(*Boolean expression*)

Remarks: Returns messages based on a Boolean expression that can include the following message properties: *Originator*, *CSI*, *ANI*, *Destination*, *DestinationLocalized*, *JobID*, *State*, *Priority*, *Completed*, or *Delivered*.

Example:

```
set oContainer =
oServer.Recipients("0").FindWhere("JobID=55")
```

Item Property

For the Container object:

Description: Returns a list of the MessageRecipient or MessageAttachment objects that are in the container.

Usage: object.**Item**

Data Type: Integer

Remarks: This property is read-only.

For the ObjectCollection object:

Description: Returns a list of elements that are in the collection.

Applies To: **ObjectCollection** object

Usage: object.**Item**

Data Type: Integer

Remarks: This property is read-only.

JobID Property

For the Message object:

Description: Returns a unique identification number assigned to the message by the Genifax application.

Usage: object.**JobID**

Data Type: Long

Remarks: This property is read-only.

For the MessageRecipient object:

Description: Returns a unique identification number assigned to the recipient by the Genifax application.

Usage: object.**JobID**

Data Type: Long

Remarks: This property is read-only.

Messages Property

Description: Returns an object collection of Message objects from the message server.

Applies To: **MessageServer** object

Usage: set [collection of objects] = object.**Messages**

Data Type: Object collection

Remarks: This property is read-only.

Example:
```
'Fill a new container with the messages
'on the queue
Dim MessageContainer As Object
Set MessageContainer = server.Messages
```

MyAssistant Property

Description: Returns the e-mail address of the user's assistant.

Applies To: **User** object

Usage: user.MyAssistant="[*e-mail address*]"

Data Type: String

Remarks: Assistants are used in routing rules.

Example:

```
Dim User As New User
  for each user in Server.Users
  user.MyAssistant="jsmith@company.com"
  user.Save
next
```

MyPrinter Property

Description: Returns the name of and path to the user's personal printer

Applies To: **User** object

Usage: user.MyPrinter="[*servername**printername*]"

Data Type: String

Remarks: Personal printers are used in routing rules.

Example:

```
Dim User As New User
  for each user in Server.Users
  user.MyPrinter="\\server2\hplaserj"
  user.Save
next
```

Originator Property

Description: Returns the e-mail address of the user who sent the message.

Applies To: **Message** object

Usage: object.**Originator**

Data Type: String

Remarks: This property is read-only.

PreviewRequired Property

Description: Returns or sets a value to indicate whether the sender wants to preview the fax before it is sent.

Applies To: **MessageRecipient** object

Usage: object.**PreviewRequired**

Data Type: Long

Priority Property

For the Message object:

Description: ***Returns or sets a value for the priority assigned to the message.

Usage: object.**Priority** [= *value*]

Data Type: Typedef Enum **PriorityType**

***One of the following enumerated values:

Value	*Constant*	*Description*
0	High	High priority
5	Normal	Medium priority
10	Low	Low priority

Remarks: High priority messages are sent first.

For the MessageRecipient object:

Description: ***Returns or sets a value for the priority assigned to the recipient.

Usage: object.**Priority** [= *value*]

Data Type: Typedef Enum **PriorityType**

***One of the following enumerated values:

Value	*Constant*	*Description*
0	High	High priority
5	Normal	Medium priority
10	Low	Low priority

Remarks: High priority messages are sent first.

Setting this property will override a **Priority** property value set from the Message object.

Example:

```
'Set BillingCode, Priority, TemplateFileName
APISampleMessage.Priority = cmbMessagePriority
APISampleMessage.BillingCode = _
 txtMessageBillingCode
APISampleMessage.TemplateFilename = _
 txtMessageCoverPage
```

RecipientID Property

Description: Returns the index value of the recipient.

Applies To: **MessageRecipient** object

Usage: object.**RecipientID**

Data Type: Long

Remarks: This property is read-only.

The index value is zero-based, relative to the parent message.

Recipients Property

For the Message object:

Description: Returns a container of MessageRecipient objects.

Usage: set [cObj] = object.**Recipients**

Data Type: Container object

Remarks: This property is read-only.

For the MessageServer object:

Description: Returns an object collection of MessageRecipient objects from the message server.

Usage: set [cVal] = object.**Recipients**

Data Type: Collection of objects

Remarks: This property is read-only.

Example:

```
Dim StatusContainer As Object
Dim RecipientStatusContainer As Object
Set StatusContainer = server.Messages
Set RecipientStatusContainer = _
 server.Recipients
```

RecipientType Property

Description: Returns or sets the type of recipient — either Genifax or Genidocs.

Applies To: **MessageRecipient** object

Usage: object.**RecipientType** [= *value*]

Data Type: Typedef Enum **ApiRecipientType**

The enumerated values are:

Value	*Constant*	*Description*
0	Fax	A Genifax message
1	Gdoc	***A Genidocs message

Sort Property

Description: Returns a container of MessageRecipient objects or an object collection of MessageRecipient objects; the objects are sorted by the property and order you specify in ascending or descending order. The property you use to sort the objects can be any valid property for the object.

Applies To: Container of **MessageRecipient** objects

Usage: oContainer.Sort "*message property*", *"ascending or descending"*

Example:

```
set oContainer =
oServer.Recipients("0").FindWhere("Originator=
john@company.com")
oContainer.Sort "JobID", "Ascending"
```

State Property

Description: Returns a value representing which state of processing the recipient is in.

Applies To: **MessageRecipient** object

Usage: object.**State**

Data Type: Long

Remarks: Compose, Dispatch, and Delivery are some examples of states.

StateText Property

Description: Returns a text string that describes which state the recipient is currently in.

Applies To: **MessageRecipient** object

Usage: object.**StateText**

Data Type: String

Remarks: This property is read-only.

Status Property

Description: Returns a value that indicates the status of the recipient in its current state.

Applies To: **MessageRecipient** object

Usage: object.**Status**

Data Type: Long

Remarks: If a message has not completed processing in a particular state, the status returned might be, for example, "Pending." If the message has completed processing within a state, the status can be either "Successful" or "Unsuccessful."

This property is read-only.

StatusText Property

Description: Returns a text string that describes the current status of the recipient.

Applies To: **MessageRecipient** object

Usage: object.**StatusText**

Data Type: String

Remarks: This property is read-only.

TemplateFilename Property

For the Message object:

Description: Returns or sets the file name of the template that will be used for the message.

Usage: object.**TemplateFilename** [= *value*]

Data Type: String

Remarks: A template determines the layout of a fax cover page and the information that it will contain (sender name, sender address, recipient fax number, and so on).

The template file must reside in the Templates directory on the Omtool Message Server. The default location created during the installation of the Omtool Message Server is:

Omtool\MessageServer\Languages\ENU\Templates

For the MessageRecipient object:

Description: Returns or sets the name of the template that will be used for the recipient.

Usage: object.**TemplateFilename** [= *value*]

Data Type: String
(See remarks and example sections.)

Remarks: A template determines the layout of a fax cover page and the information that it will contain (sender name, sender address, recipient fax number, and so on).

The template file must reside in the Templates directory on the Omtool Message Server. The default location created during the installation of the Omtool Message Server is:

Omtool\MessageServer\Languages\ENU\Templates

Setting this property will override a **TemplateFilename** property value set from the Message object.

Example:

```
'Set BillingCode, Priority, TemplateFileName
APISampleMessage.Priority = cmbMessagePriority
APISampleMessage.BillingCode = _
 txtMessageBillingCode
APISampleMessage.TemplateFilename = _
 txtMessageCoverPage
```

Methods

This section lists the Genifax COM API methods alphabetically and provides information about each one. In some cases, code snippets from the sample application are included in the Examples section of the method. To see the full context in which these snippets appear, open the code window of the sample application project and search for the desired text string.

Add Method

Description: Adds a recipient or an attachment to the Message object.

Applies To: **Message** object

Usage: object.**Add** (*pObject*)

Arguments: *pObject*
Data type: object
Either a MessageAttachment object or a MessageRecipient object.
Remarks: Before calling this method, you must create and set properties for either a new MessageAttachment object or a new MessageRecipient object.

If this method is called after a MessageAttachment object is created, it adds an attachment to the message. If it is called after a MessageRecipient object is created, it adds a recipient to the message.

Example:

```
'add the attachment to the new message
APISampleMessage.Add attachmentToFax
```

Approval Accepted Method

Description: Enables an outbound fax to be delivered to its intended destination once an approval manager reviews and approves it.

Applies To: **Recipient** object (only when the outbound fax is pending approval)

Usage: oRecipient.ApprovalAccepted

Approval Rejected Method

Description: Prevents an outbound fax from being delivered to its intended destination after an approval manager reviews and rejects it.

Applies To: **Recipient** object (only when the outbound fax is pending approval)

Usage: oRecipient.ApprovalRejected

Cancel Method

Description: Cancels the recipient so that the outbound fax cannot be delivered to its intended destination.

Applies To: **Recipient** object

Usage: oRecipient.Cancel

ConnectToServer Method

Description: Makes the initial connection to the message server.

Usage: object.**ConnectToServer** (*servername, username*)

Arguments: *servername*
Data type: String
Description:Name of the server

username
Data type: String
Description:Name of the user

Returns:A new MessageServer object.

Remarks: Messages sent will show the e-mail address of the user who is logged on.

Example:

```
If txtLoginID.Text <> "" Then
Set con = New ServConnect
On Error Resume Next
Set server = con.ConnectToServer _
(txtServerName.Text, txtLoginID.Text)
If Err.Number = 0 Then ' No error
```

ConnectToServerAdmin Method

Description: Makes the initial connection to the message server. Once connected, status can be viewed but no messages can be sent.

Applies To: **MessageRecipient** object

Usage: object.**ConnectToServerAdmin** (*servername*)

Arguments: *servername*
Data type: String
Description:Name of the server

Returns:A new MessageServer object.

Example:

```
Set con = New ServConnect
Set server = con.ConnectToServerAdmin _
 (frmConnectToServer.txtServerName.Text)
```

Delete Method

Description: Deletes a recipient from the Message object.

Applies To: **Message** object

Usage: object.**Delete**

Remarks: Messages can be deleted only when they are complete (i.e., in history).

Example:

```
'delete the message
APISampleMessage.Delete
```

GetTemplateVar Method

For the Message object:

Description: Gets a template variable from the Message object.

Usage: object.**GetTemplateVar** (*bstrName*)

Arguments: *bstrName*
Data type: String
Description:The text that appears in the *bstrName* field of the message template.

Remarks: The fax cover page is based on the template specified by the **TemplateFilename** property of the Message object.

For the MessageRecipient object:

Description: Gets a template variable from the Recipient object.

Applies To: **MessageRecipient** object

Usage: object.**GetTemplateVar** (*bstrName*)

Arguments: *bstrName*
Data type: String
Description:The text that appears in the *bstrName* field of the recipient template.

Remarks: The fax cover page is based on the template specified by the **TemplateFilename** property of the MessageRecipient object.

New Method

Description: Creates a new Container object.

Applies To: **Container** object

Usage: object.**New** ()

Arguments: *none*
Data type: Object
Description:A new Container object.

Example:
```
'Create new message object called
'APISampleMessage
Dim APISampleMessage As New Message
Set APISampleMessage = MessageContainer.New
```

Open Method

Description: Opens an object in the container.

Applies To: **Container** object

Usage: object.**Open** (*JobID*)

Arguments: *JobID*
Data type: String
Description:A unique value that identifies the object.

Returns: Either a Recipient or a Message object.

OpenAttachment Method

Description: Opens the attachment(s) of the message.

Applies To: **Message** object

Usage: object.**Open** (*JobID*)

Arguments: *JobID*
Data type: String
Description: A unique value that identifies the attachment object.

Returns: An attachment object.

OpenRecipient Method

Description: Opens the recipient(s) of the message.

Applies To: **Message** object

Usage: object.**OpenRecipient** (*JobID*)

Arguments: *JobID*
Data type: String
Description:A unique value that identifies the Recipient object.

Returns:A Recipient object.

Preview Accepted Method

Description: Enables an outbound fax to be delivered to its intended destination once the sender reviews and accepts it.

Applies To: **Recipient** object (only when the outbound fax is waiting to be previewed)

Usage: oRecipient.ReviewAccepted

Preview Rejected Method

Description: Enables an outbound fax to be delivered to its intended destination once the sender reviews and rejects it.

Applies To: **Recipient** object (only when the outbound fax is waiting to be previewed)

Usage: oRecipient.Rejected

Print Final Form Method

Description: Allows the script to print a fax on the specified printer using an optional template.

Applies To: **Recipient** object

Usage: oRecipient.PrintFinalForm "\\uncserver\printername", "printer.omtpl"

Remarks: In this example, printer.omtpl is the name of the printer template file. (The OMTPL extension indicates and Omtool template.) Leave the template value empty if you do not want to use a template.

Refresh Method

Description: Refreshes the Message and Recipient objects in the container.

Applies To: **Container** object

Usage: object.**Refresh** ()

Remarks: This eliminates the need to get the Message and Recipient objects from the MessageServer object.

Resend Method

Description: Causes a previously delivered fax to be placed in the sending queue. This method works only with items that reside in the final state in the history.

Applies To: Recipient object

Usage: oRecipient.resend

ReturnToFaxCenter Method

Description: Enables the Genifax user to send any fax that arrived from FaxCenter back to FaxCenter.

Applies To: **User** object

Usage: user.ReturnToFaxCenter="[*value*]"

Remarks: This method places an appropriate entry in the message's journal and moves the message from the user's inbound queue to the queue for FaxCenter.

Save Method

Description: Sends the Genifax message to the server for processing.

Applies To: **Message** object

Usage: object.**Save** ()

Remarks: Before this method can be called, the message must have at least one attachment and one recipient (with the **Destination** property set).

Example:

```
'Save the message (i.e. deliver it to the
'message server)
APISampleMessage.Save
```

Save Merged Final Form Method

Description: Allows the script to save a local copy of the fax in its final format— TIF or PDF. The folder you specify becomes a temporary folder and stores the filename that gets returned.

Applies To: Recipient object

Usage: sFilename = oRecipient.SaveMergedFinalForm "c:\folder", "TIF" or "PDF"

SetTemplateVar Method

For the Message object:

Description: Sets a variable on the message's template.

Usage: object.**SetTemplateVar** (*bstrName*, *bstrValue*)

Arguments: *bstrName*
Data type: string
Description:Name of the field on the message's template.

bstrValue
Data type: string
Description:Text that will be displayed in the specified field.

See Also: The **TemplateFilename** property of the Message object, which sets the name of the template that will be used.

Example:

```
'Add the Coverpage Variables to the Coverpage
APISampleMessage.SetTemplateVar "RECIP_NAME",
_
 frmMessageCheck.lstEmailaddressees.List(0)
APISampleMessage.SetTemplateVar "SenderName", _
 frmConnectToServer.txtLoginID
APISampleMessage.SetTemplateVar "COMMENTS", _
 "This space for rent"
```

For the MessageRecipient object:

Description: Sets a variable on the recipient's template.

Usage: object.**SetTemplateVar** (*bstrName*, *bstrValue*)

Arguments: *bstrName*
Data type: String
Description:Name of the field on the recipient's template.

bstrValue
Data type: String
Description:Text that will be displayed in the specified field.