Omtool COM API User Guide

For AccuRoute and Genifax environments

January 2013

Omtool, Ltd.
6 Riverside Drive
Andover, MA 01810
Phone: +1/1 978 327 5700
Toll-free in the US: +1/1 800 886 7845
Fax: +1/1 978 659 1300

Omtool Europe
25 Southampton Buildings
London
WC2A 1AL
United Kingdom
Phone: +44/0 20 3043 8580
Toll-free in the UK: +44/0 80 0011 2981
Fax: +44/0 20 3043 8581

Web: http://www.omtool.com

Omtool support and sales

Online resources

The Omtool web site provides you with 24-hour access to documentation, software updates and other downloads, and detailed technical information that can help you troubleshoot issues. Go to http://www.omtool.com/support and log in using your customer number. Then click one of the following:

- **Knowledge Base** to access technical articles.
- **Downloads & Docs** to access online documentation, software updates, and downloads.

Customer service and technical support

Contact Omtool Customer Service or Technical Support using any of the following methods:

- **Phone**: +1/1 978 327 6800 or +1/1 888 303 8098 (toll-free in the US)
- **Fax**: +1/1 978 659 1301
- **E-mail**: customerservice@omtool.com or support@omtool.com

Technical support requires an active support contract. For more information, go to http://www.omtool.com/support/entitlements.cfm.

Sales, consulting services, licenses, and training

Contact Omtool Sales using any of the following methods:

- **Phone**: +1/1 978 327 5700 or +1/1 800 886 7845 (toll-free in the US)
- **Fax**: +1/1 978 659 1300
- **E-mail**: sales@omtool.com

Contents

Section 4: Objects and collections

Section 5: Error codes

Section 6: Examples of Common Functions

Contents

Section 1: Introduction

This section includes:

Omtool COM API

The Omtool COM API is a standard application program interface that enables you to customize the functionality of your Omtool solution and integrate it with your existing technologies.

The most common questions on the Omtool COM API are:

- **What kinds of functions does the Omtool COM API support?** The Omtool COM API supports many server functions, enabling you to write applications or scripts that can connect to the message server as an administrator or user; create messages; create Embedded Directives and Routing Sheets, and associate them with messages; submit messages to the message server; create users, delegates, recipients, and attachments; and more.
- **Who should use the Omtool COM API?** Users of the Omtool COM API should be experienced application developers familiar with COM applications and the Omtool server.
- **What information can I find in this documentation?** This documentation contains requirements and installation instructions, a description of the objects and their relationships, a complete listing of the properties and methods for each object, information on using the Sample project, a complete list of error codes, examples of functions, and a list of additional resources including documentation and technical support.
- **How do I get started?** Install the Omtool COM API, run the modules in the Sample project, and review the code examples in this documentation.

Omtool server

The Omtool server, the centerpiece of any AccuRoute or Genifax implementation, is an enterprise document routing solution that integrates with existing mail systems; document management systems and records management systems; multifunction devices and other network hardware such as fax machines, scanners, and copiers; enterprise database applications; and other enterprise business applications.

It accepts inbound messages through numerous connectors and directly from client applications, and delivers outbound messages through its connectors to recipients and destinations. All messages are subject to rules, which determine the path of each message through the server's processing components. Each message has a journal where the server enters all the actions it has taken on the message.

The server maintains a set of user defaults, which are settings and permissions that apply to unknown users, and a database of registered users, which represents known users that have unique settings and permissions. Users can have delegates, assistants, and printers assigned to them, and can also be subscribed to preview, approval, or review for messages they send.

Embedded Directives

Omtool first introduced the concept of using the Embedded Directive to route messages. The Embedded Directive, an encoded key, represents a set of routing instructions.

Embedded Directives are created using the Omtool COM API, the AccuRoute Client, and the Omtool Web Client. Once users begin creating Embedded Directives, the highly intelligent Omtool server can apply an existing Embedded Directive to messages, scan messages to detect an Embedded Directive on a Routing Sheet, and apply user-specified Embedded Directives to messages.

When an Embedded Directive has been associated with a message, the Omtool server's Embedded Directive Manager component decodes the Embedded Directive and retrieves the routing instructions for the server. A single Embedded Directive can be used repeatedly to streamline common document routing practices, can be configured to expire after a particular date and time, and can be associated with individual users or a user account that a group of users can access.

Routing Sheets

The Routing Sheet is a cover sheet for a message that contains an Embedded Directive. Users can scan documents with Routing Sheets, and using the Omtool COM API, you can create messages and attach Routing Sheets.

The Routing Sheet serves two purposes:

- It contains the Embedded Directive, the encoded key that represents the routing instructions for the message.
- It functions as a cover sheet for the message. Depending on the message settings, the Routing Sheet accompanies the message to its final destination or is removed once the server decodes the Embedded Directive on it. Routing Sheets are so versatile that even users outside your company can send documents that contain Routing Sheets to your company and know that these documents are delivered reliably to the appropriate recipients and destinations.

Messages

Messages is a general term used to describe jobs on the server:

- **An inbound or outbound message associated with an Embedded Directive.**
  - Example of an inbound message - Someone outside your company sends a message via e-mail or fax to someone inside your company. The message includes a Routing Sheet that contains an Embedded Directive. (In this instance, the server must be configured appropriately to identify inbound messages with Routing Sheet, decode the Embedded Directive, and deliver the message to the recipients and destinations indicated by the Embedded Directive.)
  - Example of an outbound message - A user sends a message that includes a Routing Sheet through e-mail, a network scanner or copier, or Filescan. (In this instance, the server must be configured

appropriately to identify outbound messages with Routing Sheets, decode the Embedded Directive, and deliver the message to the recipients and destinations indicated by the Embedded Directive.)

- **An outbound message submitted to a connector.**

  - Example - A user copies a message to a Filescan folder in the local area network. The Filescan connector on the server retrieves the message. Then the Omtool server prepares and delivers the message according to the routing instructions in the control file. (In this instance, the server must be configured appropriately to process messages through Filescan.)

- **An inbound or outbound fax.**

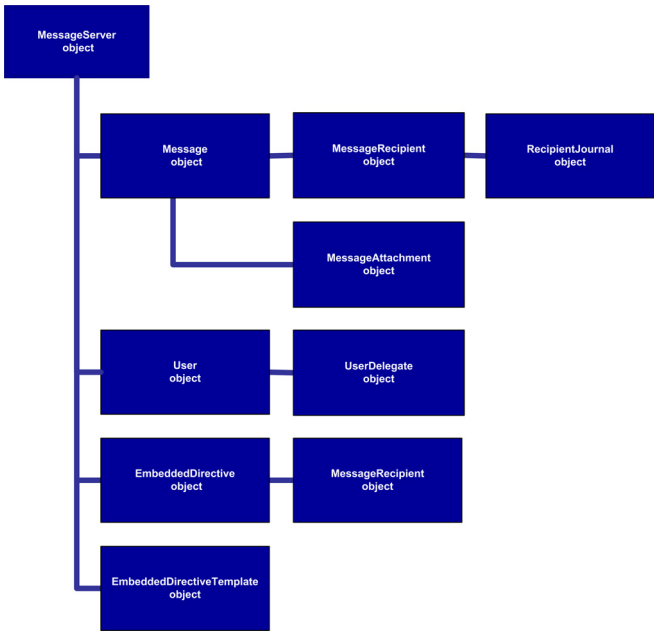
  - Example of an inbound fax - Someone outside your company sends a fax to someone inside your company. The server, in a typical configuration, queries your company's address book using the DID number or DTMF digits associated with the inbound fax, obtains the e-mail address of the recipient, and routes the fax to the recipient. (In this instance, the server must be configured appropriately to process and deliver inbound faxes.)
  - Example of an outbound fax - A user sends a fax to someone outside your company. The server prepares the message and routes it to the Telco connector. Then the Modem Server manages the delivery of the fax. (In this instance, the server must be configured appropriately to process and deliver outbound faxes.)

Omtool COM API objects

The Omtool COM API supports the following objects:

- **EmbeddedDirective** object - Represents an embedded directive. (An embedded directive is an encoded key that represents a set of routing instructions.)
- **EmbeddedDirectiveTemplate** object - Represents a routing sheet template.
- **Message** object - Represents a message on the Omtool server. (A message has at least one recipient or destination and at least one attachment.)
- **MessageAttachment** object - Represents a file attachment to a message.
- **MessageRecipient** object - Represents the recipient or destination of a message.
- **MessageServer** object - Represents the message server.
- **RecipientJournal** object - Represents the message journal. (The message journal is a log of events pertaining to the message; each time the server acts on a message, it adds an event to the journal and indicates whether the action was successful. If the action failed, the description of the event usually indicates the reason for the failure.)
- **ServConnect** object - Represents a connection to the message server.
- **User** object - Represents a user within your organization.
- **UserDelegate** object - Represents a delegate. A delegate is a user who can manage the messages of another user, and/or send messages on behalf of another user.

Figure 1-1: **Omtool COM API objects and their relationships**



The MessageServer object exposes the Message object (which exposes the MessageRecipient object and the MessageAttachment object), the EmbeddedDirective object (which exposes the MessageRecipient object), the EmbeddedDirectiveTemplate object, and the User object (which exposes the UserDelegate object).

Section 2: Installation

This section includes:

Components installed by the Omtool COM API setup

The Omtool COM API setup program installs the required DLL files and the Sample project for Visual Basic.

The required DLL files are:

- …\Program Files\Common Files\Omtool\omfGFScriptingU.DLL (registered)
- …\Program Files\Common Files\Omtool\omfGFInterfacesps.DLL (registered)
- …\Program Files\Common Files\Omtool\omimg.DLL

The Sample project for Visual Basic is installed to …\Omtool\ComAPI\Sample with several modules:

- EDProcessSample.BAS
- EDSample.BAS
- EDScanSample.BAS
- FAXSample.BAS

Installation requirements

Install the Omtool COM API on the system where you intend to run the applications you create. The system must meet the following requirements:

- Windows 2003/XP/2000
- Service account for the Omtool server must have Distributed COM access permissions on the system where you install the Omtool COM API and run applications. (This is the user account you used to install the Omtool server. It is the logon account for all Om* services.)
- Must be in the same domain as the Omtool server or in another domain that has a bidirectional trust with the server's domain.
- Any other resource requirements necessary to support the applications you create.

Omtool COM API installation

Download the Omtool COM API from the Omtool web site. Then install it on the system where you intend to run the applications you create.

To install the Omtool COM API:

1. Run the Omtool COM API update utility on the Omtool server. This installs the Omtool COM API setup to `...\Omtool\OmtoolServer\Clients\ComAPI`.
2. Navigate to `...\Omtool\OmtoolServer\Clients\ComAPI` and run `setup.exe`. The InstallShield wizard displays the **Welcome** screen.
3. Click **Next**. The **License Agreement** screen appears.
4. Read the license agreement and click **Yes** if you agree to the terms. The **Choose Destination Location** screen appears.
5. Accept the default location, or choose a new location if necessary, and click **Next**. The **Start Copying Files** screen appears.
6. Review the installation settings and click **Next**. The setup program installs the Omtool COM API and displays the InstallShield Wizard **Complete** screen.
7. Click **Finish**.

You have completed the installation of the Omtool COM API. (Go to Sample projects on 3-1.)

Section 3: Sample projects

This section includes:

The Sample project is a Visual Basic project with four modules:

- **EDProcessSample** - Creates and submits a message to the message server. The message has multiple recipients, a file attachment, and an Embedded Directive associated with it. (Go to Running the EDProcessSample module on 3-1.)
- **EDSample** - Queries the message server for all the Embedded Directives associated with a user and writes the results to an output file. (Go to Running the EDSample module on 3-2.)
- **EDScanSample** - Creates and submits a message to the message server. The message has multiple recipients and a Routing Sheet attachment. (Go to Running the EDScanSample module on 3-3.)
- **FAXSample** - Creates and submits a message to the message server. The message has multiple recipients and a file attachment. Several template variables are set on the message and on recipients. Additionally, several optional properties are set on recipients. (Go to Running the FAXSample module on 3-4.)

Running the EDProcessSample module

When you run this module, the Omtool COM API creates and submits a message to the message server. (The message has multiple recipients, a file attachment, and an Embedded Directive associated with it.) Then the Dispatch component analyzes the message and applies outbound rules.

Before running this module, review the outbound rules on the message server and create new rules if necessary. (Use the Omtool Server Administrator to view and modify rules.)

To run the EDProcessSample module:

1. Open the Sample project in Visual Basic and view the code in the module EDProcessSample.
2. Locate the section of code where the module connects to the message server. It calls the method `ConnectToServer`. Modify the method arguments. The argument `servername` should be the network name or IP address of the message server, and the argument `username` should be the e-mail address of the user who is submitting the message to the server.

   When you complete this step, the line of code should look similar to:

   ```
   Set oMsgServer = oServConnect.ConnectToServer ("OmtoolServerName",
   "jsmith@company.com")
   ```

3 Locate the section of code where the module sets the property `ApplicationTag` on the new Embedded Directive object. Modify the property value. It should be the e-mail address of the user who is submitting the message to the server.

When you complete this step, the line of code should look similar to:

```
oNewED.ApplicationTag = "jsmith@company.com"
```

4 Locate the section of code where the module creates two message recipient objects. (Search on `oNewRecipient1` if you have difficulty locating this section.) Modify the properties set on these recipient objects so that the value of the property `Destination` is a valid e-mail address, fax number, or printer location in your organization, and that the value of the property `RecipientType` accurately describes the destination. Use `FaxNumber` for fax destinations, `Email` for e-mail addresses, and `Printer` for network printers.

When you complete this step, the section of code should look similar to:

```
Dim oNewRecipient1 As New MessageRecipient
oNewRecipient1.Destination = "recipient@company.com"
oNewRecipient1.RecipientType = Email
oNewRecipient1.Priority = Normal
oNewED.Add oNewRecipient1
```

Note that this code sample adds only one message recipient object to the Embedded Directive object.

5 Locate the section of code where the module sets the property `AttachmentOriginalPathName` on the message attachment object. Set the value of this property to the file path of the attachment. If you use a local drive mapping, it must be relative to the system running the application or script.

When you complete this step, the section of code should look similar to:

```
oNewAttachment.AttachmentOriginalPathName =
"\\computer\folder\filename.ext"
```

Note that this code sample uses a UNC path rather than a local drive mapping.

6 Run the module.

Track the message using the Omtool Server Administrator. If the server delivers the message successfully, verify that the message has been delivered to its final destination.

Running the EDSample module

This module queries the message server for a list of Embedded Directives based on the value of the property `ApplicationTag`. The value of this tag is traditionally the e-mail address of the user who created the Embedded Directive. Before running the module, identify the e-mail address of a user who has created one or more Embedded Directives.

To run the EDSample module:

1. Open the Sample project in Visual Basic and view the code in the module EDSample.
2. Locate the section of code where the module connects to the message server as an administrator. It calls the method `ConnectToServerAdmin`. Modify the method argument. The argument `servername` should be the network name or IP address of the message server.

   When you complete this step, the line of code should look similar to:

   ```
   Set oMsgServer = oServConnect.ConnectToServerAdmin
   ("OmtoolServerName")
   ```
3. Locate the section of code where the module calls the method `FindWhere` on the Embedded Directive container. Modify the value of the property `ApplicationTag`. It should be the e-mail address of the user whose Embedded Directives should be returned.

   When you complete this step, the line of code should look similar to:

   ```
   Set oFindEDContainer = oEDContainer.FindWhere
   ("ApplicationTag=jsmith@company.com")
   ```
4. Run the module.

To verify success, go to ....\Omtool\ComAPI\Sample and open output.txt. You should see the results of the query.

Running the EDScanSample module

When you run this module, the Omtool COM API creates and submits a message to the message server. (The message has multiple recipients and a Routing Sheet attachment.)Then the Dispatch component analyzes the message and applies outbound rules.

Before running this module, review the outbound rules on the message server and create new rules if necessary. (Use the Omtool Server Administrator to view and modify rules.)

To run the EDScanSample module:

1. Open the Sample project in Visual Basic and view the code in the module EDScanSample.
2. Locate the section of code where the module connects to the message server. It calls the method `ConnectToServer`. Modify the method arguments. The argument `servername` should be the network name or IP address of the message server, and the argument `username` should be the e-mail address of the user who is submitting the message to the server.

   When you complete this step, the line of code should look similar to:

   ```
   Set oMsgServer = oServConnect.ConnectToServer ("OmtoolServerName",
   "jsmith@company.com")
   ```
3. Locate the section of code where the module sets the property `ApplicationTag` on the new Embedded Directive object. Modify the property value. It should be the e-mail address of the user who is submitting the message to the server.

   When you complete this step, the line of code should look similar to:

   ```
   oNewED.ApplicationTag = "jsmith@company.com"
   ```

4 Locate the section of code where the module creates two message recipient objects. (Search on `oNewRecipient1` if you have difficulty locating this section.) Modify the properties set on these recipient objects so that the value of the property `Destination` is a valid e-mail address, fax number, or printer location in your organization, and that the value of the property `RecipientType` accurately describes the destination. Use `FaxNumber` for fax destinations, `Email` for e-mail addresses, and `Printer` for network printers.

   When you complete this step, the section of code should look similar to:

```
Dim oNewRecipient1 As New MessageRecipient
oNewRecipient1.Destination = "recipient@company.com"
oNewRecipient1.RecipientType = Email
oNewRecipient1.Priority = Normal
oNewED.Add oNewRecipient1
```

   Note that this code sample adds only one message recipient object to the Embedded Directive object.

5 Locate the section of code where the module sets the property `TemplateFilename` on the message attachment object. Verify that the value of this property is the filename of a cover page template that resides on the message server in ...\Omtool\OmtoolServer\Languages\xxx\Templates.

   When you complete this step, the section of code should look similar to:

```
oNewMessage.TemplateFilename = "OmtoolCoverpage"
```

6 Locate the section of code where the module sets the property `AttachmentOriginalPathName` on the message attachment object. Set the value of this property to the file path of the attachment. If you use a local drive mapping, it must be relative to the system running the application or script.

   When you complete this step, the section of code should look similar to:

```
oNewAttachment.AttachmentOriginalPathName =
"\\computer\folder\filename.ext"
```

   Note that this code sample uses a UNC path rather than a local drive mapping.

7 Run the module.

Track the message using the Omtool Server Administrator. If the server delivers the message successfully, verify that the message has been delivered to its final destination.

Running the FAXSample module

When you run this module, the Omtool COM API creates and submits a message to the message server. (The message has multiple recipients and a file attachment.) Then the Dispatch component analyzes the message and applies outbound rules.

Before running this module, review the outbound rules on the message server and create new rules if necessary. (Use the Omtool Server Administrator to view and modify rules.)

To run the FAXSample module:

1. Open the Sample project in Visual Basic and view the code in the module FAXSample.
2. Locate the section of code where the module connects to the message server. It calls the method `ConnectToServer`. Modify the method arguments. The argument `servername` should be the network name or IP address of the message server, and the argument `username` should be the e-mail address of the user who is submitting the message to the server.

   When you complete this step, the line of code should look similar to:

   ```
   Set oMsgServer = oServConnect.ConnectToServer ("OmtoolServerName",
   "jsmith@company.com")
   ```

3. Locate the section of code where the module sets sender template variables on the message object. (Search on `SENDER_EMAIL` if you have difficulty locating this section.) Modify the variables as desired so that the values describe the message sender.

   When you complete this step, the section of code should look similar to:

   ```
   oNewMessage.SetTemplateVar "SENDER_EMAIL", "jsmith@company.com"
   oNewMessage.SetTemplateVar "SENDER_NAME", "John Smith"
   oNewMessage.SetTemplateVar "SENDER_BUSINESS_PHONE", "328-1465"
   oNewMessage.SetTemplateVar "SENDER_BUSINESS_FAX_PHONE", "890-1957"
   oNewMessage.SetTemplateVar "SENDER_MAILING_ADDRESS", "8 Longshore
   Drive, Suite D"
   ```

4. Locate the section of code where the module creates three message recipient objects. (Search on `oNewRecipient1` if you have difficulty locating this section.) Modify the properties set on these recipient objects so that the value of the property `Destination` is a valid e-mail address, fax number, or printer location in your organization, and that the value of the property `RecipientType` accurately describes the destination. Use `FaxNumber` for fax destinations, `Email` for e-mail addresses, and `Printer` for network printers.

   When you complete this step, the section of code should look similar to:

   ```
   Dim oNewRecipient1 As New MessageRecipient
   oNewRecipient1.Destination = "recipient@company.com"
   oNewRecipient1.RecipientType = Email
   ```

   Note that this code sample adds only one message recipient object to the message object.

5. Locate the section of code where the module sets recipient template variables on the recipient objects. (Search on `RECIP_EMAIL` if you have difficulty locating this section.) Modify the variables as desired so that the values describe the message recipient.

   When you complete this step, the section of code should look similar to:

   ```
   oNewRecipient1.SetTemplateVar "RECIP_EMAIL",
   "tjones@recipcompany.com"
   oNewRecipient1.SetTemplateVar "RECIP_NAME", "Tom Jones"
   oNewRecipient1.SetTemplateVar "RECIP_BUSINESS_PHONE", "627-3920"
   oNewRecipient1.SetTemplateVar "RECIP_MAILING_ADDRESS", "16399 West
   56th"
   ```

   Note that this code sample defines the recipient variables for only one recipient.

6 Locate the section of code where the module sets the property `TemplateFilename` on the message object. Verify that the value of this property is the filename of a cover page template that resides on the message server in `...\Omtool\OmtoolServer\Languages\xxx\Templates`.

When you complete this step, the section of code should look similar to:

```
oNewMessage.TemplateFilename = "OmtoolCoverpage"
```

7 Locate the section of code where the module sets the property `AttachmentOriginalPathName` on the message attachment object. Set the value of this property to the file path of the attachment. If you use a local drive mapping, it must be relative to the system running the application or script.

When you complete this step, the section of code should look similar to:

```
oNewAttachment.AttachmentOriginalPathName =
"\\computer\folder\filename.ext"
```

Note that this code sample uses a UNC path rather than a local drive mapping.

8 Locate the section of code where the module creates two notification recipient objects. (Search on `oNotifRecipient1` if you have difficulty locating this section.) Modify the properties set on these recipient objects so that the value of the property `Destination` is a valid e-mail address, fax number, or printer location in your organization, and that the value of the property `RecipientType` accurately describes the destination. Use `FaxNumber` for fax destinations, `Email` for e-mail addresses, and `Printer` for network printers.

When you complete this step, the section of code should look similar to:

```
Dim oNotifRecipient1 As New NotificationRecipient
oNotifRecipient1.Destination = "sender@mail.com"
oNotifRecipient1.RecipientType = Email
oNewMessage.Add oNotifRecipient1
```

Note that this code sample adds only one notification recipient object to the message object.

9 Run the module.

Track the message using the Omtool Server Administrator. If the server delivers the message successfully, verify that the message has been delivered to its final destination.

Section 4: Objects and collections

This section includes:

Collection

Collections support EmbeddedDirective objects, Message objects, MessageRecipient objects, User objects, and UserDelegate objects.

Properties

EntryID

Data type: string, read-only

Description: Returns the entry ID number of an object in the collection.

Usage: `object.EntryID`

Item

Data type: ret collection object

Description: Returns an object based on its enumeration value within the collection.

Usage: `collection.Item`

Count

Data type: short, read-only

Description: Returns the number of items in the collection.

Usage: `collection.Count`

Methods

FindWhere

Description: Searches the collection and returns the objects that match the criteria.

Usage: `collection.FindWhere (boolExpression)`

Argument: `boolExpression`

- **Description:** Description: Boolean expression that sets the criteria for the query on the collection, for example: `object.FindWhere ("Email=jsmith@company.com")`
- **Remarks:**
  - On a collection of User objects or UserDelegate objects, use the Email property. On a collection of MessageRecipient objects, use one of the following properties: ANI, CancelPending, Completed, CSI, Delivered, Destination, DestinationLocalized, JobID, Originator, Priority, RecipientType, State, or Status.
  - The following operators are valid within a FindWhere expression: `and`, `or`, `( )`, `=`, `< >`, `>`, and `<`.
- **Data type:** string

New

Description: Returns a new collection object.

Usage: `object.New`

Open

Description: Returns an existing collection object.

Usage: `object.Open (EntryID)`

Argument: `EntryID`

- **Description:** Entry ID number of the object
- **Data type:** string

Refresh

Description: Refreshes the collection.

Usage: `object.Refresh`

Sort

Description: Sorts objects in the collection.

Usage: `collection.Sort (bstrProperty, bstrOrder)`

Argument: `bstrProperty`

- **Description:** Property by which the collection objects should be sorted
- **Data type:** string

Argument: `bstrOrder`

- **Description:** Use ascending or descending for the sort order
- **Data type:** string

EmbeddedDirective

For properties and methods pertaining to a collection of EmbeddedDirective objects, go to Collection (4-1).

Properties

ApplicationName

Data type: string

Description: Returns or sets the name of the application that creates the Embedded Directive. (Note that AccuRoute Desktop, the AccuRoute Client, the Omtool Web Client, and the Omtool COM API can all generate Embedded Directives.)

Usage/read: `oEmbeddedDirective.ApplicationName`

Usage/write: `oEmbeddedDirective.ApplicationName = [bstr value]`

ApplicationTag

Data type: string

Description: Returns or sets the application tag. Omtool client applications set the ApplicationTag property using the e-mail address of the originator, but you can use any type of string data that might be helpful to your organization. If you do not use the e-mail address of the originator, Omtool recommends that you implement your application tag scheme consistently.

Usage/read: `oEmbeddedDirective.ApplicationTag`

Usage/write: `oEmbeddedDirective.ApplicationTag = [bstr value]`

BillingCode

Data type: string

Description: Returns or sets the billing code associated with the Embedded Directive.

This property can be set on MessageRecipient objects; however, when you set this property on the EmbeddedDirective object, you can access the property readily when you open the EmbeddedDirective object instead of having to open individual MessageRecipient objects and read the property from each one.

Usage/read: `oEmbeddedDirective.BillingCode`

Usage/write: `oEmbeddedDirective.BillingCode = [bstr value]`

Comment

Data type: string

Description: Returns or sets the comment intended for recipients.

The comment appears on the cover page or Routing Sheet, if applicable, when the template file includes the variable %COMMENT%.

A comment can be set on MessageRecipient objects via the SetTemplateVar and GetTemplateVar methods; however, when you set this property on the EmbeddedDirective object, you can access the property readily when you open the EmbeddedDirective object instead of having to open individual MessageRecipient objects and read or write the comment.

Usage/read: `oEmbeddedDirective.Comment`

Usage/write: `oEmbeddedDirective.Comment = [bstr value]`

Created

Data type: date, read-only

Description: Returns the date when the Embedded Directive was created.

Usage: `oEmbeddedDirective.Created`

EntryID

Data type: string, read-only

Description: Returns the entry ID number of the Embedded Directive.

Usage: `oEmbeddedDirective.EntryID`

Expires

Data type: date

Description: Returns or sets an expiration date on the Embedded Directive.

When the Embedded Directive expires, it becomes invalid. If the invalid Embedded Directive still exists on the message server, meaning that it hasn't been deleted by the cleanup process, you change the expiration date on the Embedded Directive object and make the Embedded Directive valid again.

This value appears on the cover page or Routing Sheet, if applicable, when the template file includes the variable %ROUTINGSHEET_DATE_EXPIRES%.

Usage/read: `oEmbeddedDirective.Expires`

Usage/write: `oEmbeddedDirective.Expires = [date value]`

Recipients

Data type: ret collection, read-only

Description: Returns a collection of MessageRecipient objects.

Usage: `oEmbeddedDirective.Recipients`

RemoveRoutingSheetForDelivery

Data type: bool

Description: Returns or sets a boolean value that determines whether the Routing Sheet is removed from the message prior to delivery. When this value is true, the message server removes the Routing Sheet prior to delivery.

Usage/read: `oEmbeddedDirective.RemoveRoutingSheetForDelivery`

Usage/write: `oEmbeddedDirective.RemoveRoutingSheetForDelivery = [bool value]`

SingleUse

Data type: bool

Description: Returns or sets a boolean value that determines whether the Embedded Directive can be used more than once. When this value is true and the Embedded Directive has been used, the message server retains the Embedded Directive and marks it as used.

This value appears on the cover page or Routing Sheet, if applicable, when the template file includes the variable %ROUTINGSHEET_SINGLE_USE%.

Usage/read: `oEmbeddedDirective.SingleUse`

Usage/write: `oEmbeddedDirective.SingleUse = [bool value]`

StyleEntryID

Data type: string

Description: Returns or sets the style entry ID number of the Routing Sheet template.

When composing a Routing Sheet, the message server uses the style entry ID to identify the Routing Sheet template on the message server. (All Routing Sheet templates must reside on the message server in `…\Omtool\OmtoolServer\Languages\xxx\EmbeddedDirectives\`.)

If you want to set this property on an Embedded Directive object, use the EntryID property on the EmbeddedDirectiveTemplate object to return the style entry ID of the Routing Sheet template. Then use this return value to set the StyleEntryID property on the Embedded Directive object.

Usage/read: `oEmbeddedDirective.StyleEntryID`

Usage/write: `oEmbeddedDirective.StyleEntryID = [bstr value]`

Subject

Data type: string

Description: Returns or sets the subject of the Embedded Directive.

The subject appears on the cover page or Routing Sheet, if applicable, when the template file includes the variable %SUBJECT%.

When this property is set on a MessageRecipient object, it overrides the property that has been set on the EmbeddedDirective object.

Usage/read: `oEmbeddedDirective.Subject`

Usage/write: `oEmbeddedDirective.Subject = [bstr value]`

Title

Data type: string

Description: Returns or sets the title of the Embedded Directive.

The title appears on the cover page or Routing Sheet, if applicable, when the template file includes the variable %ROUTINGSHEET_TITLE%.

Usage/read: `oEmbeddedDirective.Title`

Usage/write: `oEmbeddedDirective.Title = [bstr value]`

Version

Data type: string, read-only

Description: Returns the version number of the Embedded Directive. (This property is used internally by the message server to identify the version of the software that created the Embedded Directive.)

Usage: `oEmbeddedDirective.Version`

Methods

Add

Description: Adds a MessageRecipient object to the Embedded Directive.

Usage: `oEmbeddedDirective.Add (oMessageRecipient)`

Argument: `oMessageRecipient`

- **Description:** MessageRecipient object
- **Data type:** object

ComposeRoutingSheet

Data type: ret string

Description: Composes a Routing Sheet based on the Embedded Directive and returns the filename as a string value.

Usage: `oEmbeddedDirective.ComposeRoutingSheet (bstrFolderToSaveTo)`

Argument: `bstrFolderToSaveTo`

- **Description:** Recipient object or attachment object
- **Data type:** string

Save

Description: Saves or updates an Embedded Directive object.

Before calling this method, verify that the Embedded Directive has at least one recipient.

Usage: `oEmbeddedDirective.Save`

EmbeddedDirectiveTemplate

An EmbeddedDirective Template object represents a Routing Sheet template on the message server.

Routing Sheet templates reside on the message server in `...\Omtool\OmtoolServer\Languages\xxx\Templates`. A Routing Sheet template file can any of the following file types: OMTPL, DOC, RTF, HTM, or HTML.

Each Routing Sheet template file can be accompanied by a WTX file with the same filename. (For example, the Routing Sheet template OmtoolRoutingSheet.OMTPL can be accompanied by OmtoolRoutingSheet.WTX.) The WTX file contains a description of the template.

Properties

Description

Data type: string, read-only

Description: Returns the description of the Routing Sheet template. The description comes from the WTX file. If no WTX file exists, the description is the filename of the Routing Sheet template.

Usage: `oEmbeddedDirectiveTemplate.Description`

EntryID

Data type: string, read-only

Description: Returns the entry ID number that identifies the Routing Sheet template on the message server. Use the return value to set the StyleEntryID property on an Embedded Directive object.

Usage: `oEmbeddedDirectiveTemplate.EntryID`

Name

Data type: string, read-only

Description: Returns the filename of the Routing Sheet template.

Usage: `oEmbeddedDirectiveTemplate.Name`

Methods

There are no methods for this object.

Message

Some properties of this object are also associated with other objects. For example, the property BillingCode is a property of the Message object and the MessageRecipient object.

When a property is set on the super-object, it propagates to the sub-object. For example, when the property BillingCode is set on a Message object, it propagates to all MessageRecipient objects associated with the message.

However, when a property has been set on both the super-object and the sub-object, the property set on the sub-object prevails for that object. For example, the property BillingCode is set as value A on the Message object and as value B on the MessageRecipient object. The MessageRecipient object assumes value B.

For properties and methods pertaining to a collection of Message objects, go to Collection (4-1).

Properties

AccessCode

Data type: string

Description: Returns or sets the access code on the message object.

Usage/read: `oMessage.AccessCode`

Usage/write: `oMessage.AccessCode = [bstr value]`

Attachments

Data type: ret collection, read-only

Description: Returns a collection of MessageAttachment objects.

Usage: `oMessage.Attachments`

BillingCode

Data type: string

Description: Returns or sets the billing code on the message object.

Usage/read: `oMessage.BillingCode`

Usage/write: `oMessage.BillingCode = [bstr value]`

DateCompleted

Data type: date, read-only

Description: Returns the date when the message server finished processing the message.

Usage: `oMessage.DateCompleted = [date value]`

DateSendAfter

Data type: date

Description: Returns or sets the date and time when the server can attempt to deliver the message. This property can be used to delay the delivery of messages.

This property can be set only before the message object has been saved.

Usage/read: `DateSendAfter`

Usage/write: `oMessage.DateSendAfter = [date value]`

DateSubmitted

Data type: date, read-only

Description: Returns the date and time the message was submitted to the message server for processing.

Usage: `oMessage.DateSubmitted`

EntryID

Data type: string, read-only

Description: Returns the entry ID number of the message.

Usage: `oMessage.EntryID`

JobID

Data type: long, read-only

Description: Returns the job ID number of the message.

Usage: `oMessage.JobID`

Notification

Data type: NotificationType

Description: Returns or sets the notification type on the message object. Valid values are:

- `NotifyOnFailure` or `1` - Send a notification message only if the message server fails the message.
- `NotifyOnSuccess` or `2` - Send a notification message only if the message server delivers the message.
- `NotifyOnBoth` or `3` - Send a notification message when the message server delivers or fails the message.

The message server sends notification messages to the originator and to any other notification recipients associated with the message.

When this property is not set on a message object, the message server does not send notification messages.

Usage/read: `oMessage.Notification`

Usage/write: `oMessage.Notification = [NotificationType]`

Originator

Data type: string, read-only

Description: Returns the e-mail address of the message sender.

Usage: `oMessage.Originator`

Priority

Data type: PriorityType

Description: Returns or sets the priority level on the message object.

Valid values are `normal`, `high`, and `low`, or `0` (High), `5` (Normal), and `10` (Low).

Usage/read: `oMessage.Priority`

Usage/write: `oMessage.Priority = [PriorityType]`

Recipients

Data type: ret collection, read-only

Description: Returns a collection of MessageRecipient objects.

Usage: `oMessage.Recipients`

SecureFax

Data type: bool

Description: Returns or sets a boolean value that determines whether secure delivery is enabled on the message object. When the value is true, the message server delivers the message to fax recipients using secure delivery.

The secure delivery feature requires configuration on the message server. If secure delivery is not configured and enabled, the message server fails the message. Note that secure delivery requires a fax recipient to complete and return a registration form that entitles the recipient to receive secure faxes automatically.

Usage/read: `oMessage.SecureFax`

Usage/write: `oMessage.SecureFax = [bool value]`

TemplateFilename

Data type: string

Description: Returns or sets the filename of the cover page template that the message server should use when composing the cover page.

Cover page template files reside on the message server in `...\Omtool\OmtoolServer\Languages\xxx\Templates`.

Usage/read: `oMessage.TemplateFilename`

Usage/write: `oMessage.TemplateFilename = [bstr value]`

Methods

Add

Description: Adds a MessageAttachment object or MessageRecipient object to the message.

Usage: `oMessage.Add (object)`

Argument: `object`

- **Description:** MessageAttachment object or MessageRecipient object
- **Data type:** object

AddRecipientToExisting

Description: Adds a MessageRecipient object to an existing message.

Usage: `oMessage.AddRecipientToExisting (oMessageRecipient)`

Argument: `oMessageRecipient`

- **Description:** MessageRecipient object
- **Data type:** object

Delete

Description: Deletes the message object. The message server must have finished processing the message in order for this method to be valid.

Usage: `oMessage.Delete`

GetTemplateVar

Description: Retrieves the value of a template variable (string) set on the message object.

Usage: `oMessage.GetTemplateVar (bstrName)`

Argument: `bstrName`

- **Description:** Name of a template variable set on the message object
- **Data type:** string

OpenAttachment

Description: Returns a MessageAttachment object.

Usage: `oMessage.OpenAttachment (EntryID)`

Argument: `EntryID`

- **Description:** Entry ID number of the message
- **Data type:** string

OpenRecipient

Description: Returns a MessageRecipient object.

Usage: `oMessage.OpenRecipient (EntryID)`

Argument: `EntryID`

- **Description:** Entry ID number of the recipient
- **Data type:** string

Save

Description: Saves or updates a message object. The message must have at least one recipient and one attachment in order for this method to be valid.

Usage: `oMessage.Save`

SetTemplateVar

Description: Sets a variable on the message object.

Usage: `oMessage.SetTemplateVar (bstrName, bstrVal)`

Argument: `bstrName`

- **Description:** Name of a template variable (do not include percent symbols)
- **Data type:** string

Argument: `bstrVal`

- **Description:** Value of the template variable
- **Data type:** string

MessageAttachment

All message content is stored as attachments to the message. This includes documents that comprise the body of the message, templates that are used to compose the message, and plain text that comprises the e-mail body of e-mail messages.

Properties

Required on message attachment objects: AttachmentOriginalPathName

AttachmentOriginalPathName

Data type: string

Description: Returns or sets the path to the message attachment. Must be a UNC path or local drive mapping relative to the system running the application or script. Must include the filename and extension.

Usage/read: `oMessageAttachment.AttachmentOriginalPathName`

Usage/write: `oMessageAttachment.AttachmentOriginalPathName = [bstr value]`

AttachmentSize

Data type: long, read-only

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

Usage: `oMessageAttachment.AttachmentSize`

EntryID

Data type: string, read-only

Description: Returns the entry ID number of the message attachment.

Usage: `oMessageAttachment.EntryID`

IsEmailBody

Data type: bool

Description: Returns or sets a boolean value that determines whether the attachment should be used as the body of an e-mail message. When the value is true and the recipient destination is an e-mail address, the message server inserts the file content into the body of the e-mail message.

Usage/read: `oMessageAttachment.IsEmailBody`

Usage/write: `oMessageAttachment.IsEmailBody = [bool value]`

IsTemplate

Data type: bool, read-only

Description: Returns or sets a boolean value that indicates whether an attachment is a template such as a notification template, Routing Sheet template, cover page template, etc. When the value is true, the message attachment is a template.

Usage: `oMessageAttachment.IsTemplate`

Methods

Delete

Description: Deletes the MessageAttachment object.

Usage: `oMessageAttachment.Delete`

SaveAs

Description: Saves the MessageAttachment object to a specified location.

Usage: `oMessageAttachment.SaveAs (bstrPathname)`

Argument: `bstrPathname`

- **Description:** Complete path to the file destination including the filename (UNC path or a local drive mapping relative to the system running the application or script)
- **Data type:** string

SaveFinalFormAs

Description: Saves a copy of the MessageAttachment object to a specified location in the specified format.

Usage: `oMessageAttachment.SaveFinalFormAs (bsPathname, bsFinalForm)`

Argument: `bsPathname`

- **Description:** Complete path to the file destination including the filename (UNC path or a local drive mapping relative to the system running the application or script)
- **Data type:** string

Argument: `bsFinalForm`

- **Description:** File format in which the message should be saved (G4.TIF, PDF, OCR.PDF, OCR.DOC, OCR.RTF, OCR.TXT, or *)
- **Data type:** string

MessageRecipient

Some properties of this object are also associated with other objects. For example, the property BillingCode is a property of the Message object and the MessageRecipient object.

When a property is set on the super-object, it propagates to the sub-object. For example, when the property BillingCode is set on a Message object, it propagates to all MessageRecipient objects associated with the message.

However, when a property has been set on both the super-object and the sub-object, the property set on the sub-object prevails for that object. For example, the property BillingCode is set as value A on the Message object and as value B on the MessageRecipient object. The MessageRecipient object assumes value B.

For properties and methods pertaining to a collection of MessageRecipient objects, go to Collection.

Properties

Required on message recipient objects: Destination, RecipientType

AccessCode

Data type: string

Description: Returns or sets the access code Some telephone systems require users to provide an access code; the access code associates the call with a user or group of users. Access codes are frequently used for billing and call tracking purposes. on the message recipient object.

Usage/read: `oMessageRecipient.AccessCode`

Usage/write: `oMessageRecipient.AccessCode = [bstr value]`

ANI

Data type: string, read-only

Description: Returns the caller ID of the sender. The property ANI pertains to inbound faxes.

Usage: `oMessageRecipient.ANI`

ApprovalRequested

Data type: bool

Description: Returns or sets a boolean value that determines whether the message requires approval before the message server can deliver it to the message recipient. When the value is true, approval was requested or required on the message that the message server delivers to the message recipient. (Note that the value does not indicate whether the message was approved or rejected, or whether the message is pending approval.)

Usage/read: `oMessageRecipient.ApprovalRequested`

Usage/write: `oMessageRecipient.ApprovalRequested = [bool value]`

Attachments

Data type: ret collection, read-only

Description: Returns a collection of MessageAttachment objects.

Usage: `oMessageRecipient.Attachments`

BillingCode

Data type: string

Description: Returns or sets the billing code A billing code is a numeric string that associates messages with a particular account or recipient. Billing codes are used to recover costs associated with sending and receiving documents. on the message recipient object.

Usage/read: `oMessageRecipient.BillingCode`

Usage/write: `oMessageRecipient.BillingCode = [bstr value]`

CancelPending

Data type: bool, read-only

Description: Returns a boolean value that determines whether the message has been cancelled. When the value is true, the message has been cancelled and the message server does not deliver it to the message recipient.

Usage: `oMessageRecipient.CancelPending`

Completed

Data type: long, read-only

Description: Returns a value that indicates whether the message is in its final state. For example: The job has completed

Usage: `oMessageRecipient.Completed`

Composed

Data type: bool, read-only

Description: Returns a boolean value that indicates whether the message has been composed. When the value is true, the message has been composed by the message server. (Note that the message server composes inbound and outbound messages.)

Usage: `oMessageRecipient.Composed`

CSI

Data type: string, read-only

Description: Returns the identity of the channel that received the inbound fax.

Usage: `oMessageRecipient.CSI`

DateCompleted

Data type: date, read-only

Description: Returns the date and time when the message server finishes processing the message.

This property is valid only when the message is complete. Use the Completed property to determine whether the message is complete.

Usage: `oMessageRecipient.DateCompleted`

DateReceived

Data type: date, read-only

Description: Returns the date and time when the message server received the message.

Usage: `oMessageRecipient.DateReceived`

DateRouted

Data type: date, read-only

Description: Returns the date and time when the message server routed the message from FaxCenter to the recipient.

Usage: `oMessageRecipient.DateRouted`

DateSendAfter

Data type: date

Description: Returns or sets the date and time when the server can deliver the message to the recipient. This property can be used to delay the delivery of a message.

Usage/read: `oMessageRecipient.DateSendAfter`

Usage/write: `oMessageRecipient.DateSendAfter = [date value]`

DateSubmitted

Data type: date, read-only

Description: Returns the date and time when the user submitted the message to the message server.

Usage: `oMessageRecipient.DateSubmitted`

Delivered

Data type: bool, read-only

Description: Returns a boolean value that indicates whether the message server delivered the message successfully. When this value is true, the message server has delivered the message successfully.

Usage: `oMessageRecipient.Delivered`

Destination

Data type: string

Description: Returns or sets the destination for the recipient.

Note that the property Destination must always be accompanied by the property RecipientType, which enables the message server to interpret the destination address correctly. For example:

```
oMessageRecipient.Destination = "jsmith@company.com"
oMessageRecipient.RecipientType = Email
```

Usage/read: `oMessageRecipient.Destination`

Usage/write: `oMessageRecipient.Destination = [bstr value]`

DestinationLocalized

Data type: string, read-only

Description: Returns the localized destination address of the recipient. The message server localizes fax addresses during processing.

Note that the return value can contain overrides specified by Dispatch component rules.

Usage: `oMessageRecipient.DestinationLocalized`

Duration

Data type: string, read-only

Description: Returns the length of the telephone call during which the fax was transmitted.

Usage: `oMessageRecipient.Duration`

EntryID

Data type: string, read-only

Description: Returns the entry ID number of the recipient.

Usage: `oMessageRecipient.EntryID`

FaxCenterGenerated

Data type: bool, read-only

Description: Returns a boolean value that indicates whether the fax was generated by FaxCenter. When the value is true, the message was generated by FaxCenter, meaning that the inbound message was routed to FaxCenter and then routed to the recipient.

Usage: `oMessageRecipient.FaxCenterGenerated`

FaxCenterRouted

Data type: bool, read-only

Description: Returns a boolean value that indicates whether the message server routed a FaxCenter-generated message. When this value is true, the message server routed the FaxCenter-generated message to the message recipient. A false value signifies that the message server has not routed the FaxCenter-generated message to the message recipient, or that the message is not a FaxCenter-generated message.

Usage: `oMessageRecipient.FaxCenterRouted`

FinalFormCode

Data type: string

Description: Returns or sets the file extension associated with the file type that should be delivered to the recipient.

If you set this property, the rule that routes messages to the delivering connector (for example, the SMTP connector for an e-mail message) must allow the message sender to override the delivery format specified in the rule. (This option is called Allow sender to override the Delivery Format.)

Examples of valid final form codes include:

- `G4.TIF` - Results in a G4 TIF file.
- `PDF` - Results in a PDF file.
- `OCR.PDF` - Results in a PDF file that preserves the existing text or contains OCR-rendered text from the original input file.

- `OCR.DOC` - Results in a Word document that contains the OCR-rendered text from the original input file.
- `OCR.RTF` - Results in a rich text format document that contains the OCR-rendered text from the original input file.
- `OCR.TXT` - Results in a simple text document that contains the OCR-rendered text from the original input file.
- `*` - Results in the original input file.

Usage/read: `oMessageRecipient.FinalFormCode`

Usage/write: `oMessageRecipient.FinalFormCode = [bstr value]`

Inbound

Data type: bool, read-only

Description: Returns a boolean value that indicates whether a message was received or delivered by the Telco connector. When this value is true, the message is an inbound fax that has been received in its entirety or an outbound fax.

Usage: `oMessageRecipient.Inbound`

IsANotification

Data type: bool, read-only

Description: Returns a boolean value that indicates whether the message is a notification message generated by the message server. When this value is true, the message is a notification message generated by the message server.

Usage: `oMessageRecipient.IsANotification`

JobID

Data type: long, read-only

Description: Returns the job ID number of the message.

Usage: `oMessageRecipient.JobID`

JournalEntries

Data type: ret collection, read-only

Description: Returns a collection of RecipientJournal objects.

Usage: `oMessageRecipient.JournalEntries`

Notification

Data type: NotificationType

Description: Returns or sets the notification type for the message.

- `NotifyOnFailure` or `1` - Send a notification message only if the message server fails the message.
- `NotifyOnSuccess` or `2` - Send a notification message only if the message server delivers the message.

- `NotifyOnBoth` or `3` - Send a notification message when the message server delivers or fails the message.

The message server sends notification messages to the originator and to any other notification recipients associated with the message.

When this property is not set on a message object, the message server does not send notification messages.

Usage/read: `oMessageRecipient.Notification`

Usage/write: `oMessageRecipient.Notification = [NotificationType]`

NumberOfPages

Data type: string, read-only

Description: Returns the total number of pages in the message. Valid for faxes only.

Usage: `oMessageRecipient.NumberOfPages`

Originator

Data type: string, read-only

Description: Returns the e-mail address of the sender.

Usage: `oMessageRecipient.Originator`

PreviewRequired

Data type: bool

Description: Returns or sets a boolean value that determines whether preview When preview is required on a message, the sender must review and approve the message before the message server can deliver it. If the sender does not approve the message, the message server stops processing the message immediately. has been requested. When the value is true, the message sender requested a message preview. (Note that the value does not indicate whether the sender has reviewed the message or whether preview is pending.)

Usage/read: `oMessageRecipient.PreviewRequired`

Usage/write: `oMessageRecipient.PreviewRequired = [bool value]`

PreviewURL

Data type: string, read-only

Description: Returns the URL of the message preview.

Usage: `oMessageRecipient.PreviewURL`

PrintedByWebSite

Data type: bool

Description: Returns or sets a boolean value that indicates whether the message has been printed from the Omtool Web Client. When the value is true, the message recipient printed the message using the Omtool Web Client.

Usage/read: `oMessageRecipient.PrintedByWebSite`

Usage/write: `oMessageRecipient.PrintedByWebSite = [bool value]`

Priority

Data type: PriorityType

Description: Returns or sets the priority level for the recipient. Valid values are normal, high, and low, or 0 (High), 5 (Normal), and 10 (Low).

Usage/read: `oMessageRecipient.Priority`

Usage/write: `oMessageRecipient.Priority = [PriorityType]`

RecipientID

Data type: long, read-only

Description: Returns the index value of a recipient. The value is zero-based relative to the message.

Usage: `oMessageRecipient.RecipientID`

RecipientType

Data type: ApiRecipientType

Description: Returns or sets the recipient type on the MessageRecipient object.

Valid values are:

- `Email` or `0` - Indicates that the destination is an e-mail address.
- `FaxNumber` or `1` - Indicates that the destination is a fax number.
- `Printer` or `2` - Indicates that the destination is a network printer. When using this value to set the property RecipientType, the property Destination should be set to the IP address or UNC path of the network printer.
- `DID` or `3` - Indicates that the destination is a DID number. (This value occurs with inbound faxes only. Do not use this value when setting the property RecipientType on a MessageRecipient object.)
- `UNC` or `5` - Indicates that the destination is a UNC path. When using this value to set the property RecipientType, the property Destination should be set to the UNC path where the message should be delivered and saved.
- `CSI` or `6` - Indicates the identity of the fax device or channel that transmitted the inbound fax. (This value is set by the Telco connector and occurs with inbound faxes only. Do not use this value when setting the property RecipientType on a MessageRecipient object.)
- `ANI` or `7` - Indicates the ANI Automatic Number Identification, commonly known as caller ID. Your organization's telephone service provider likely offers this feature as part of your telephone service plan. of the telephone line that transmitted the inbound fax. (This value is set by the Telco connector and occurs with inbound faxes only. Do not use this value when setting the property RecipientType on a MessageRecipient object.)
- `TSI` or `8` - Indicates the identity of the fax device or channel that received the inbound fax. (This value is set by the Telco connector and occurs with inbound faxes only. Do not use this value when setting the property RecipientType on a MessageRecipient object.)
- `EmbeddedDirective` or `16` - Indicates that an Embedded Directive determines the recipient type on the MessageRecipient object. When using this value to set the property RecipientType, the property Destination should be set to oEmbeddedDirective.ID. where oEmbeddedDirective is the name of the Embedded Directive object.

When this property is not set, the message server uses the default value `FaxNumber`.

Note that the property RecipientType accompanies the property Destination. It enables the message server to interpret the destination address correctly. For example:

```
oMessageRecipient.Destination = "jsmith@company.com"
oMessageRecipient.RecipientType = Email
```

Usage/read: `oMessageRecipient.RecipientType`

Usage/write: `oMessageRecipient.RecipientType = [ApiRecipientType]`

SecureFax

Data type: bool

Description: Returns or sets a boolean value that determines whether secure delivery is enabled on the message object. When the value is true, the message server delivers the message to fax recipients using secure delivery.

The secure delivery feature requires configuration on the message server. If secure delivery is not configured and enabled, the message server fails the message. Note that secure delivery requires a fax recipient to complete and return a registration form that entitles the recipient to receive secure faxes automatically.

Usage/read: `oMessageRecipient.SecureFax`

Usage/write: `oMessageRecipient.SecureFax = [bool value]`

State

Data type: long, read-only

Description: Returns the numerical code that represents the state of the message.

Usage: `oMessageRecipient.State`

StateText

Data type: string, read-only

Description: Returns a description of the message state. Examples of states are Compose and Delivery.

Usage: `oMessageRecipient.StateText`

Status

Data type: long, read-only

Description: Returns a numerical code that represents the status of the message in its current state.

Usage: `oMessageRecipient.Status`

StatusText

Data type: string, read-only

Description: Returns a description of the status. Examples are Pending, Successful, and Failed.

Usage: `oMessageRecipient.StatusText`

Subject

Data type: string

Description: Returns or sets the subject of the message.

The subject appears on the cover page or Routing Sheet, if applicable, when the template file includes the variable %SUBJECT%.

When this property is set on an EmbeddedDirective object and a MessageRecipient object, the value set on the MessageRecipient object prevails for that message recipient.

Usage/read: `oMessageRecipient.Subject`

Usage/write: `oMessageRecipient.Subject = [bstr value]`

TemplateFilename

Data type: string

Description: Returns or sets the filename of the cover page template that the message server should use when composing the cover page. Note that cover page template files reside on the message server in `...\Omtool\OmtoolServer\Languages\xxx\Templates`.

Usage/read: `oMessageRecipient.TemplateFilename`

Usage/write: `oMessageRecipient.TemplateFilename = [bstr value]`

ViewedByWebSite

Data type: bool

Description: Returns or sets a boolean value that indicates whether the message has been viewed from the Omtool Web Client. When the value is true, the message recipient viewed the message using the Omtool Web Client.

Usage/read: `oMessageRecipient.ViewedByWebSite`

Usage/write: `oMessageRecipient.ViewedByWebSite = [bool value]`

Methods

ApprovalAccepted

Description: Enables the message server to continue processing the message for the message recipient after an approval manager approves the message.

Usage: `oMessageRecipient.ApprovalAccepted`

ApprovalRejected

Description: Prevents the message server from delivering the message to the message recipient after an approval manager rejects the message.

Usage: `oMessageRecipient.ApprovalRejected (bstrReason)`

Argument: `bstrReason`

- **Description:** Comments for the sender that explain why the message has been rejected
- **Data type:** string

Cancel

Description: Cancels the MessageRecipient object; this prevents the message server from delivering the message to the message recipient.

Usage: `oMessageRecipient.Cancel`

Delete

Description: Deletes the MessageRecipient object. Valid only when the message is in its final state.

Usage: `oMessageRecipient.Delete`

FindWhere

Data type: ret collection objects

Description: Searches the collection of MessageRecipient objects and returns the objects that match the criteria.

Usage: `oMessageRecipient.FindWhere (boolExpression)`

Argument: `boolExpression`

- **Description:** Boolean expression that sets the criteria for the query on the collection, for example: oMessageRecipient.FindWhere ("Originator=jsmith@company.com")
- **Remarks:** This method can be used with only one of the following properties: ANI, CancelPending, Completed, CSI, Delivered, Destination, DestinationLocalized, JobID, Originator, Priority, RecipientType, State, or Status. The following operators are valid within a FindWhere expression: and, or, ( ), =, < >, >, and <.
- **Data type:** string

GetTemplateVar

Description: Retrieves the value of a template variable (string) set on the message recipient object.

Usage: `oMessageRecipient.GetTemplateVar (bstrName)`

Argument: `bstrName`

- **Description:** Name of a template variable set on the recipient object
- **Data type:** string

LogError

Description: Adds an error to the message journal.

Usage: `oMessageRecipient.LogError (bstrDesc, lError)`

Argument: `bstrReason`

- **Description:** Short description of the error
- **Data type:** string

Argument: `lError`

- **Description:** Error code
- **Data type:** long

LogInformation

Description: Adds an entry to the message journal.

Usage: `oMessageRecipient.LogInformation (bstrDesc)`

Argument: `bstrDesc`

- **Description:** Entry text
- **Data type:** string

PreviewAccepted

Description: Enables the message server to continue processing the message for the message recipient after the sender accepts the preview.

Usage: `oMessageRecipient.PreviewAccepted`

PreviewDeclined

Description: Prevents the message server from delivering the message to the message recipient after the sender rejects the preview.

Usage: `oMessageRecipient.PreviewDeclined`

Print

Description: Prints the message on a shared printer in the network. This method allows you to specify the template that should accompany the message and the final form of the message sent to the printer.

Usage: `oMessageRecipient.Print (bstrPrinterName, bstrTemplate, bstrFinalForm)`

Argument: `bstrPrinterName`

- **Description:** Identity of the printer
- **Remarks:** Use a UNC path (\\printserver\printer) or IP address (IP:172.16.6.33). With an IP address, the Xerox Walk-Up Printing Driver must be install on the message server. Additionally,

the OmtoolPS printer must be configured to use this driver. For more information on locating and installing the driver, consult the Omtool server installation guide.

- **Data type:** string

Argument: `bstrTemplate`

- **Description:** Filename (without the file extension) of the Routing Sheet template or cover page template that should accompany the message
- **Data type:** string

Argument: `bstrFinalForm`

- **Description:** Final form that should be sent to the printer (G4.TIF, PDF, OCR.PDF, OCR.DOC, OCR.RTF, OCR.TXT, or *)
- **Data type:** string

PrintFinalForm

Description: Prints the message on a shared printer in the network. This method allows you to specify the template that should accompany the message.

Usage: `oMessageRecipient.PrintFinalForm (bstrPrinterName, bstrTemplate)`

Argument: `bstrPrinterName`

- **Description:** Identity of the printer
- **Remarks:** Use a UNC path (\\printserver\printer) or IP address (IP:172.16.6.33). With an IP address, the Xerox Walk-Up Printing Driver must be install on the message server. Additionally, the OmtoolPS printer must be configured to use this driver. For more information on locating and installing the driver, consult the Omtool server installation guide.
- **Data type:** string

Argument: `bstrTemplate`

- **Description:** Filename (without the file extension) of the Routing Sheet template or cover page template that should accompany the message
- **Data type:** string

Resend

Description: Recalls a previously sent message and puts it in the state delivery. Valid only when the message is complete. (Use the Completed property to determine whether the message is complete.)

Usage: `oMessageRecipient.Resend`

ReturnToFaxCenter

Description: Enables the message recipient to return the message to FaxCenter. (The message recipient might return a message to FaxCenter if the message should be sent to a different recipient.)

Usage: `oMessageRecipient.ReturnToFaxCenter`

SaveMergedFinalForm

Description: Saves a copy of all message attachments to the specified location, with the specified filename, and in the specified format.

Usage: `oMessageRecipient.SaveMergedFinalForm (bstrPath, bstrFinalForm, bstrFileName)`

Argument: `bstrPath`

- **Description:** Complete path to the file destination excluding the filename (UNC path or local drive mapping relative to the system running the application or script)
- **Data type:** string

Argument: `bstrFinalForm`

- **Description:** Format of the file to be saved (G4.TIF, PDF, OCR.PDF, OCR.DOC, OCR.RTF, OCR.TXT, or *)
- **Data type:** string

Argument: `bstrFileName`

- **Description:** Name of the file to be saved (excluding the file extension)
- **Data type:** string

SetTemplateVar

Description: Sets a variable on the recipient object.

Usage: `oMessageRecipient.SetTemplateVar (bstrName, bstrVal)`

Argument: `bstrName`

- **Description:** Name of a template variable (without percent symbols)
- **Data type:** string

Argument: `bstrVal`

- **Description:** Value of the template variable
- **Data type:** string

MessageServer

Properties

Delegates

Data type: ret collection, read-only

Description: Returns a collection of UserDelegate objects.

Usage: `oMessageServer.Delegates`

EmbeddedDirectives

Data type: ret collection, read-only

Description: Returns a collection of EmbeddedDirective objects.

Usage: `oMessageServer.EmbeddedDirective`

EmbeddedDirectivesTemplates

Data type: ret collection, read-only

Description: Returns a collection of EmbeddedDirectiveTemplate objects.

Usage: `oMessageServer.EmbeddedDirectivesTemplates`

Messages

Data type: ret collection, read-only

Description: Returns a collection of Message objects.

Usage: `oMessageServer.Messages`

Recipients

Data type: ret collection, read-only

Description: Returns a collection of MessageRecipient objects.

Usage: `oMessageServer.Recipients`

User

Data type: ret object, read-only

Description: Returns a User object. Valid when connected to the message server in user mode.

Usage: `oMessageServer.User`

Users

Data type: ret collection, read-only

Description: Returns a collection of User objects; the collection includes all registered users. Valid when connected to the message server in administrator mode.

Usage: `oMessageServer.Users`

Methods

There are no methods for this object.

RecipientJournal

Properties

Date

Data type: date, read-only

Description: Returns the date of an event in the message journal.

Usage: `oRecipientJournal.Description`

Description

Data type: string, read-only

Description: Returns the description of an event in the message journal.

Usage: `oRecipientJournal.Description`

ErrorCode

Data type: long, read-only

Description: Returns the error code associated with an event in the message journal.

Usage: `oRecipientJournal.ErrorCode`

Sequencer

Data type: long, read-only

Description: Returns the sequence number of an event in the journal.

This property can be useful in sorting journal events in the order they occurred. Event order begins with the lowest value.

Usage: `oRecipientJournal.Sequencer`

Methods

There are no methods for this object.

ServConnect

Properties

There are no properties for this object.

Methods

ConnectToServer

Description: Makes a connection to the message server in user mode and returns a MessageServer object. In user mode, the application or script can access only the user's messages and Embedded Directives in their respective container objects.

Usage: `oServConnect.ConnectToServer (servername, username)`

Argument: `servername`

- **Description:** Network name or IP address of the message server
- **Data type:** string

Argument: `username`

- **Description:** E-mail address of the user (also the originator e-mail address associated with all messages and Embedded Directives created; the message server uses this e-mail address to send notification messages)
- **Data type:** string

ConnectToServerAdmin

Description: Makes a connection to the message server in administrator mode and returns a MessageServer object. In administrator mode, the application or script can access all messages, Embedded Directives, and users in their respective container objects.

Usage: `oServConnect.ConnectToServer (servername)`

Argument: `servername`

- **Description:** Network name or IP address of the message server
- **Data type:** string

User

For properties and methods pertaining to a collection of User objects, go to Collection (4-1).

Properties

Some of these properties pertain to features that require additional configuration on the message server. For example, the features preview, approval, and review require component-level configuration and user-level configuration. Additionally, FaxCenter requires user-level configuration.

ApprovalManagers

Data type: ret collection, read-only

Description: Returns a collection of User objects; these objects represent the approval managers for the user.

Usage: `oUser.ApprovalManagers`

ApprovalSubordinates

Data type: ret collection, read-only

Description: Returns a collection of User objects; these objects represent other users whose faxes the user can approve.

Usage: `oUser.ApprovalSubordinates`

CanApproveFaxes

Data type: bool

Description: Returns or sets a boolean value that determines whether a user can approve the faxes of other users. When the value is true, the user is an approval manager.

Usage/read: `oUser.CanApproveFaxes`

Usage/write: `oUser.CanApproveFaxes = [bool value]`

Email

Data type: string

Description: Returns or sets the e-mail address of the user.

Usage/read: `oUser.Email`

Usage/write: `oUser.Email = [bstr value]`

EnableFaxManagerCompose

Data type: bool

Description: Returns or sets a boolean value that determines whether the user can create faxes using the Omtool Web Client. When the value is true, the user can create faxes using the Omtool Web Client.

Usage/read: `oUser.EnableFaxManagerCompose`

Usage/write: `oUser.EnableFaxManagerCompose = [bool value]`

FaxManagerDelegates

Data type: ret collection, read-only

Description: Returns a collection of UserDelegate objects; these objects represent other users who are delegates to the user.

Usage: `oUser.FaxManagerDelegates`

FaxManagerSubordinates

Data type: ret collection, read-only

Description: Returns a collection of User objects; these objects represent other users to which the user is a delegate.

Usage: `oUser.FaxManagerSubordinates`

MyAssistant

Data type: string

Description: Returns or sets the assistant on a User object. The assistant, when associated with a user, receives a copy of all messages the user sends and receives.

Usage/read: `oUser.MyAssistant`

Usage/write: `oUser.MyAssistant = [bstr value]`

MyPrinter

Data type: string

Description: Returns or sets the printer on a User object. The printer, when associated with a user, generates a copy of all messages the user sends and receives.

Usage/read: `oUser.MyPrinter`

Usage/write: `oUser.MyPrinter = [bstr value]`

Name

Data type: string

Description: Returns or sets the name of the user represented by the User object.

Usage/read: `oUser.Name`

Usage/write: `oUser.Name = [bstr value]`

RoutingTypes

Data type: ret collection, read-only

Description: Returns a collection of RoutingType objects; these objects represent the routing types that are available to the user in the AccuRoute Client.

Usage: `oUser.RoutingTypes`

Methods

FindWhere

Description: Searches a collection of User objects and returns the objects that match the criteria.

Usage: `oUser.FindWhere (boolExpression)`

Argument: `boolExpression`

- **Description:** Boolean expression that sets the criteria for the query on the object collection, for example: oUser.FindWhere ("Email=jsmith@company.com")
- **Remarks:** This method can be used with the Email property only. The following operators are valid within a FindWhere expression: and, or, ( ), =, < >, >, and <.
- **Data type:** string

UserDelegate

For properties and methods pertaining to a collection of UserDelegate objects, go to Collection (4-1).

Properties

Email

Data type: string

Description: Returns or sets the e-mail address of the primary user.

Usage/read: `oUserDelegate.Email`

Usage/write: `oUserDelegate.Email = [bstr value]`

ProxyEmail

Data type: string, read-only

Description: Returns the e-mail address of the delegate user.

Usage: `oUserDelegate.ProxyEmail`

SendOnBehalf

Data type: bool

Description: Returns or sets a boolean value that determines when the delegate can send messages on behalf of another user. When the value is true, the delegate can send messages on behalf of the primary user.

Usage/read: `oUserDelegate.SendOnBehalf`

Usage/write: `oUserDelegate.SendOnBehalf = [bool value]`

ViewFaxStatus

Data type: bool

Description: Returns or sets a boolean value that determines whether the delegate can manage the messages of another user. When the value is true, the delegate can manage the messages of the primary user.

Usage/read: `oUserDelegate.ViewFaxStatus`

Usage/write: `oUserDelegate.ViewFaxStatus = [bool value]`

Method

Delete

Description: Deletes a UserDelegate object.

Usage: `oUserDelegate.Delete`

FindWhere

Description: Searches the collection of UserDelegate objects and returns the objects that match the criteria.

Usage: `oUser.FindWhere (boolExpression)`

Argument: `boolExpression`

- **Description:** Boolean expression that sets the criteria for the query on the object collection, for example: oUserDelegate.FindWhere ("Email=jsmith@company.com")
- **Remarks:** This method can be used with the Email property only. The following operators are valid within a FindWhere expression: and, or, ( ), =, < >, >, and <.
- **Data type:** string

Section 5: Error codes

This section includes:

> *Tip* If you are viewing this documentation electronically and you know the error code or error string, you can search the document using this information.

General errors

Table 5-1: **Errors related to general issues**

String	Hex code	Decimal code	Description
notLoggedIn	0x8004f064	-2147159964	You must be logged in before attempting any methods.
invalidStore	0x8004f066	-2147159962	The store is invalid.
internalConsistency	0x8004f067	-2147159961	An internal consistency check failed.
invalidFlag	0x8004f068	-2147159960	An invalid flag was specified in the method call.
badEntryID	0x8004f069	-2147159959	The supplied entry ID is not valid.
wrongEntryID	0x8004f06a	-2147159958	You may not open the requested entry ID using this container.
cantOpenEntryID	0x8004f06b	-2147159957	Failed to open the requested entry ID.
noPermission	0x8004f06c	-2147159956	You do not have permission to perform the requested operation.
notSupported	0x8004f06d	-2147159955	The object does not support that operation.
noSuchAttachment	0x8004f06e	-2147159954	The specified attachment does not exist.
failedCreateTempFile	0x8004f06f	-2147159953	Failed to create a temporary file for the operation.
invalidArgument	0x8004f070	-2147159952	An invalid argument was supplied.
objectNotSaved	0x8004f071	-2147159951	The object must be saved before this operation can be completed.
updateQfailed	0x8004f072	-2147159950	Failed to update the queue.
qLockFailed	0x8004f073	-2147159949	Failed to lock the queue.
errorsReturned	0x8004f074	-2147159948	The operation was only partially successful.
queueOpenFailed	0x8004f075	-2147159947	Failed to open the requested queue.
mustSpecifyUser	0x8004f076	-2147159946	This operation requires that you supply an e-mail name.
imageProcessError	0x8004f077	-2147159945	Failed to process the image as requested.

Table 5-1: Errors related to general issues

String	Hex code	Decimal code	Description
invalidUser	0x8004f078	-2147159944	The specified user does not exist and cannot be auto-enrolled.
autoEnrollFailure	0x8004f079	-2147159943	Failed to automatically enroll the user.
mustSpecifyOriginator	0x8004f07a	-2147159942	This operation requires that you supply an originator.
mustSpecifyRecipient	0x8004f07b	-2147159941	This operation requires that you specify a recipient.
objectMissing	0x8004f07c	-2147159940	The requested object is missing; it may have been deleted.
missingProperty	0x8004f07d	-2147159939	The object is missing a required property.
invalidPropertyValue	0x8004f07e	-2147159938	The specified value for the required property is invalid.
duplicateObject	0x8004f07f	-2147159937	This object duplicates an existing object and may not be saved.
databaseException	0x8004f080	-2147159936	An internal exception occurred while accessing the database.
invalidMessageType	0x8004f081	-2147159935	This consumer cannot handle this message type.
LockFailed	0x8004f082	-2147159934	Failed to obtain synchronization lock for shared data object.
EnumOutOfRange	0x8004f083	-2147159933	The value specified for an enum is out of range.
InvalidCoverSpecified	0x8004f084	-2147159932	The specified cover page template cannot be found.
NothingToSend	0x8004f085	-2147159931	Must specify either a cover page or attachments.
RequiresRestart	0x8004f086	-2147159930	The operation cannot be carried out. Restart the object.
CannotDeleteRecipientInProgress	0x8004f087	-2147159929	You cannot delete an active recipient. You must cancel that recipient first.
UserCancel	0x8004f088	-2147159928	The message was cancelled by the user.
unlicensedFeature	0x8004f089	-2147159927	System does not support license feature requested

Table 5-1: Errors related to general issues

String	Hex code	Decimal code	Description
CheckEventLog	0x8004f08a	-2147159926	General failure. See the event viewer for details.
profileForbidsPreview	0x8004f08f	-2147159921	This message cannot be sent. You do not have permission to request Fax Preview.
profileForbidsStatus	0x8004f090	-2147159920	This message cannot be sent. You do not have permission to request a status message.
profileMustSpecifyBillingCode	0x8004f091	-2147159919	This message cannot be sent without a billing code.
profileMustSpecifyAccessCode	0x8004f092	-2147159918	This message cannot be sent without an access code.
profileMustSpecifyCoverpage	0x8004f093	-2147159917	This message cannot be sent without a cover page.
profileMissing	0x8004f094	-2147159916	The profile assigned to you no longer exists. Until the problem with the profiles is resolved, you will not be able to send any messages.
SendOnBehalfNotAllowed	0x8004f098	-2147159912	You are not authorized to send on the behalf of the specified user.
OneOffNotAllowed	0x8004f099	-2147159911	Sending a Genifax without using the Genifax Form is not allowed.
invaldFormsOverlay	0x8004f09a	-2147159910	The specified forms overlay file is invalid.
Component_Lookup_Timeout			Lookup timeout.

Connector errors

Table 5-2: **Errors related to connectors**

String	Hex code	Decimal code	Description
TelcoBusy	0x8004f258	-2147159464	The destination number is busy.
TelcoNotFax	0x8004f259	-2147159463	The destination number is not a fax machine.
TelcoFailure	0x8004f25a	-2147159462	An unexpected error occurred in the Telco connector.
TelcoBadNumber	0x8004f25b	-2147159461	The destination number is invalid.
TelcoNoDialTone	0x8004f25c	-2147159460	The channel did not get the dial tone.
TelcoNoAnswer	0x8004f25d	-2147159459	The destination number did not answer.
TelcoConnectivity	0x8004f25e	-2147159458	There may be a hardware problem with the telco board.
TelcoDisconnect	0x8004f25f	-2147159457	The transmission was disconnected while in progress.
TelcoTransmission	0x8004f260	-2147159456	The transmission failed.
TelcoCallCollision	0x8004f261	-2147159455	The attempt to dial collided with an incoming call.
TelcoShutdown	0x8004f262	-2147159454	The channel has been shut down.
TelcoNegotiationFailed	0x8004f263	-2147159453	The initial negotiation with the dialed fax machine failed.
TelcoSpecialTones	0x8004f264	-2147159452	Special tones were received and not understood.
TelcoUnauthorized	0x8004f265	-2147159451	Not authorized to perform the required function.
TelcoCallRejected	0x8004f266	-2147159450	The call was rejected.
TelcoNoSendChannels	0x8004f267	-2147159449	None of the Telco channels are enabled for sending.
TelcoNoCarrier	0x8004f268	-2147159448	Modem did not detect a carrier.
TelcoModemFailure	0x8004f269	-2147159447	General modem hardware failure.
TelcoTimeout	0x8004f26a	-2147159446	Connection timeout.
TelcoUnderrun	0x8004f26b	-2147159445	Modem underrun.

Table 5-2: Errors related to connectors

String	Hex code	Decimal code	Description
TelcoFileSystemError	0x8004f26c	-2147159444	Modem file I/O error.
PrinterInvalid	0x8004f26c	-2147159444	The specified printer could not be found.
FolderInvalid	0x8004f280	-2147159424	The specified folder could not be found.
DestOverideNotSet	0x8004f281	-2147159423	Destination override needs to be set for filescan routing rule.
RequiredFieldNotFound	0x8004f28a	-2147159414	A required database field was not found.
DatabaseFieldsNotRead	0x8004f28b	-2147159413	The database fields could not be read.
InvalidADOFileMade	0x8004f28c	-2147159412	The ADO connector generated an output file that is corrupted.
InvalidOutDir	0x8004f28d	-2147159411	The output path specified by the ADO connector could not be written to.
FailedBillingCodeAuthorization	0x8004f294	-2147159404	One of the enabled Billing code fields was empty.
ToManyHops	0x8004f29e	-2147159394	The routing for the specified recipient resulted in too many network hops.
CircularHopLoop	0x8004f29f	-2147159393	Routing to the destination server would result in circular routing.
NotConfigAsTarget	0x8004f2a0	-2147159392	Attempted to route to a SSR connector that is not configured to receive messages.
SSRNotRunning	0x8004f2a1	-2147159391	An attempt was made to route to a remote system on which the SSR Connector is not running.
XMLConstructionError	0x8004f2a2	-2147159390	An Error occurred constructing the XML Stream for the message.
CancelledGetNextMessage	0x8004f2bc	-2147159364	The Connector Manager has requested that the producer return immediately from a blocking GetNextMessage call.
CancelledGetNextMessageFailed	0x8004f2bd	-2147159363	Failed to cancel GetNextMessage in the connector.

Table 5-2: Errors related to connectors

String	Hex code	Decimal code	Description
ProviderNotAvailable	0x8004f2be	-2147159362	The specified connector is "known", but is not enabled or is experiencing problems.
NoSuchProvider	0x8004f2bf	-2147159361	The specified provider is not known to the Connector Manager.
GetNextMessageWaitFailed	0x8004f2c0	-2147159360	Internal failure in connector GetNextMessage.
GetNextMessageIndeterminateErro r	0x8004f2c1	-2147159359	Internal failure in connector GetNextMessage.
BadABPointer	0x8004f2c2	-2147159358	An invalid address book pointer has been discovered.
MAPIAPICallFailed	0x8004f2c3	-2147159357	An internal MAPI call has failed in an unanticipated way.
EmptyPropertySet	0x8004f2c4	-2147159356	An internal configuration structure was supplied empty.
ProviderRunning	0x8004f2c5	-2147159355	The connector is already running.
AttchmentMethodNotSupported	0x8004f2c6	-2147159354	The message has an attachment that is not supported.
CouldNotDetermineAttchmentMet hod	0x8004f2c7	-2147159353	The message has an attachment that cannot be retrieved.
SubscriptOutOfRange	0x8004f2c8	-2147159352	An internal check failed.
FailedToAddProperty	0x8004f2c9	-2147159351	Failed to add a property to an internal data structure.
ExchangeConnectorThrewExceptio n	0x8004f2ca	-2147159350	The Exchange connector caught an exception and is returning a failure condition.
FailedToLoadExchangeConfiguration FunctionPointers	0x8004f2cb	-2147159349	Exchange Extension has failed to load the function pointers from the Exconncfg dll.
FailedToLoadExchangeConfiguration DLL	0x8004f2cc	-2147159348	The Exchange Extension has failed to load the Exconncfg dll.
Invalid_Email_Recipient	0x8004f2cd	-2147159347	The specified recipient is invalid or not known to the connector.
NotesConnectorThrewException	0x8004f2ce	-2147159346	The Notes Connector caught an exception and is failing gracefully.
ConnectorFailedToDeliverMessage	0x8004f2cf	-2147159345	Notes Connector failed to create message in its database.

Table 5-2: Errors related to connectors

String	Hex code	Decimal code	Description
FailedToInitNotesThread	0x8004f2d0	-2147159344	Notes Connector failed to initialize a Notes thread.
FailedToAddProducerToConfigMap	0x8004f2d1	-2147159343	Connector failed to add a Producer to the global configuration map.
FailedToStartLookupThread	0x8004f2d2	-2147159342	Failed to start Lookup thread.
FailedToLookupConsumerConfigObject	0x8004f2d3	-2147159341	Failed to find previously allocated consumer config object in the global config map.
FailedToLocateNotesMailServer	0x8004f2d4	-2147159340	Failed to find location for Notes mail server in notes.ini file.
FailedToSetNotesMailServer	0x8004f2d5	-2147159339	Failed to set location for Notes mail server in config item.
FailedToOpenNotesAddressBook	0x8004f2d6	-2147159338	Failed to open the Notes server's address book.
FailedToOpenNotesCollection	0x8004f2d7	-2147159337	Failed to open a Notes Collection.
FailedToOpenNotesDatabase	0x8004f2d8	-2147159336	Failed to open a Notes Database.
FailedToFindNotesView	0x8004f2d9	-2147159335	Failed to find a particular Notes database view.
FailedToOpenNotesMessage	0x8004f2da	-2147159334	Failed to get a handle to a Notes Message.
FailedToLookupNotesServer	0x8004f2db	-2147159333	Failed to find server in a Notes Address book.
FailedToGetLDAPHostName	0x8004f2dc	-2147159332	Failed to get the field that contains the LDAPHostName in Notes Address Book.
FailedToLookupForeignDomain	0x8004f2dd	-2147159331	Failed to look up the specified foreign domain in the Notes Database.
DuplicateForeignDomainFound	0x8004f2de	-2147159330	Duplicate entries found in address book for the specified foreign domain.
InvalidDomainRecord	0x8004f2df	-2147159329	Foreign domain record did not contain a mail file entry.
FailedToLoadFaxProfiles	0x8004f2e0	-2147159328	Failed to load fax profiles from the database.
InvalidSearchFilter	0x8004f2e1	-2147159327	Invalid search filter specified for Lookup.

Table 5-2: Errors related to connectors

String	Hex code	Decimal code	Description
FailedToInitializeLDAPServer	0x8004f2e2	-2147159326	Failed to initialize connection to LDAP.
FailedToBindToLDAPServer	0x8004f2e3	-2147159325	Failed to bind to LDAP server.
FailedToDisconnectFromLDAPServer	0x8004f2e4	-2147159324	Failed to disconnect from LDAP server.
FailedToFindFirstLDAPEntry	0x8004f2e5	-2147159323	Failed to find first LDAP record.
FailedToFindNextLDAPEntry	0x8004f2e6	-2147159322	Failed to find next LDAP record.
LDAPSearchFailed	0x8004f2e7	-2147159321	LDAP Search failed.
LDAPGetValueFailed	0x8004f2e8	-2147159320	LDAP GetValue failed.
FailedToRetrieveMessageSubject	0x8004f2e9	-2147159319	Failed to get subject from message.
FailedToRetrieveSenderProperties	0x8004f2ea	-2147159318	Failed to retrieve the sender properties from the message.
FailedToLookupSenderProfile	0x8004f2eb	-2147159317	Failed to look up sender profile.
FailedToExtractBodyIntoCDFFormat	0x8004f2ec	-2147159316	Failed to extract message body into CDF format.
FailedToExtractAttachment	0x8004f2ed	-2147159315	Failed to extract attachment from message.
FailedToUpdateFaxState	0x8004f2ee	-2147159314	Failed to update Fax State in Notes Message.
FailedToRetrieveGatewayMonitorObject	0x8004f2ef	-2147159313	Failed to retrieve the Gateway Monitor object.
FailedToStartGatewayMonitorThread	0x8004f2f0	-2147159312	Failed to start the gateway monitor thread.
FailedToDeleteNotesMessage	0x8004f2f1	-2147159311	Failed to delete a Notes message.
FailedToOpenNotesMessageFile	0x8004f2f2	-2147159310	Failed to open the Notes message file.
FailedToSendNonDeliveryReport	0x8004f2f3	-2147159309	Failed to Send a non-delivery report.
FailedToOpenMailboxFile	0x8004f2f4	-2147159308	Failed to open Notes Mailbox file.
FailedUserProfileDoesNotHaveSufficientPrivs	0x8004f2f5	-2147159307	The sender's user profile does not have this privilege set.
FailedNoUserProfileItem	0x8004f2f6	-2147159306	The sender's user profile does not have this privilege.

Table 5-2: Errors related to connectors

String	Hex code	Decimal code	Description
ClusterNotificationPortCreationFailure	0x8004f2f7	-2147159305	Failed to create cluster notification port.
ClusterFailedToOpen	0x8004f2f8	-2147159304	Failed to open the specified cluster resource.
ClusterThrewException	0x8004f2f9	-2147159303	Cluster processing caused an exception to be thrown.
RecipientTypeNotLicensed	0x8004f2fa	-2147159302	The requested recipient type cannot be used (not licensed).
RecipientFaxNumberNotFound	0x8004f2fb	-2147159301	Lookup failed to find the recipient's fax number.
RecipientNotFoundInLookup	0x8004f2fc	-2147159300	Lookup failed to find the recipient.
StrongAuthRequired	0x8004f2fd	-2147159299	Strong authentication is required.
InappropriateAuth	0x8004f2fe	-2147159298	Authentication is inappropriate.
InsufficientRights	0x8004f2ff	-2147159297	The user has insufficient access rights.
InvalidCredentials	0x8004f300	-2147159296	The supplied credential is invalid.
AuthMethodNotSupported	0x8004f301	-2147159295	The authentication method is not supported.
LDAPResultsTooLarge	0x8004f302	-2147159294	Results returned are too large.
LDAPInvalidDNSyntax	0x8004f303	-2147159293	Invalid distinguished name syntax.
UnknownOLEAttachment	0x8004f304	-2147159292	The OLE attachment is unknown and cannot be processed.
ServiceNotInitialized	0x8004f305	-2147159291	The service is not initialized.
NoRecipientDIDLookups	0x8004f306	-2147159290	There are no recipient DID lookups.

Compose errors

Table 5-3: **Errors related to the Compose process**

String	Hex code	Decimal code	Description
notLoggedIn	0x8004f064	-2147159964	You must be logged in before attempting any methods.
invalidStore	0x8004f066	-2147159962	The store is invalid.
internalConsistency	0x8004f067	-2147159961	An internal consistency check failed.
invalidFlag	0x8004f068	-2147159960	An invalid flag was specified in the method call.
badEntryID	0x8004f069	-2147159959	The supplied entry ID is not valid.
wrongEntryID	0x8004f06a	-2147159958	You may not open the requested entry ID using this container.
cantOpenEntryID	0x8004f06b	-2147159957	Failed to open the requested entry ID.
noPermission	0x8004f06c	-2147159956	You do not have permission to perform the requested operation.
notSupported	0x8004f06d	-2147159955	The object does not support that operation.
noSuchAttachment	0x8004f06e	-2147159954	The specified attachment does not exist.
failedCreateTempFile	0x8004f06f	-2147159953	Failed to create a temporary file for the operation.
invalidArgument	0x8004f070	-2147159952	An invalid argument was supplied.
objectNotSaved	0x8004f071	-2147159951	The object must be saved before this operation can be completed.
updateQfailed	0x8004f072	-2147159950	Failed to update the queue.
qLockFailed	0x8004f073	-2147159949	Failed to lock the queue.
errorsReturned	0x8004f074	-2147159948	The operation was only partially successful.
queueOpenFailed	0x8004f075	-2147159947	Failed to open the requested queue.
mustSpecifyUser	0x8004f076	-2147159946	This operation requires that you supply an e-mail name.
imageProcessError	0x8004f077	-2147159945	Failed to process the image as requested.

Table 5-3: Errors related to the Compose process

String	Hex code	Decimal code	Description
invalidUser	0x8004f078	-2147159944	The specified user does not exist and cannot be auto-enrolled.
autoEnrollFailure	0x8004f079	-2147159943	Failed to automatically enroll the user.
mustSpecifyOriginator	0x8004f07a	-2147159942	This operation requires that you supply an originator.
mustSpecifyRecipient	0x8004f07b	-2147159941	This operation requires that you specify a recipient.
objectMissing	0x8004f07c	-2147159940	The requested object is missing; it may have been deleted.
missingProperty	0x8004f07d	-2147159939	The object is missing a required property.
invalidPropertyValue	0x8004f07e	-2147159938	The specified value for the required property is invalid.
duplicateObject	0x8004f07f	-2147159937	This object duplicates an existing object and may not be saved.
databaseException	0x8004f080	-2147159936	An internal exception occurred while accessing the database.
invalidMessageType	0x8004f081	-2147159935	This consumer cannot handle this message type.
LockFailed	0x8004f082	-2147159934	Failed to obtain synchronization lock for shared data object.
EnumOutOfRange	0x8004f083	-2147159933	The value specified for an enum is out of range.
InvalidCoverSpecified	0x8004f084	-2147159932	The specified cover page template cannot be found.
NothingToSend	0x8004f085	-2147159931	Must specify either a cover page or attachments.
RequiresRestart	0x8004f086	-2147159930	The operation cannot be carried out. Restart the object.
CannotDeleteRecipientInProgress	0x8004f087	-2147159929	You cannot delete an active recipient. You must cancel that recipient first.
UserCancel	0x8004f088	-2147159928	The message was cancelled by the user.
unlicensedFeature	0x8004f089	-2147159927	System does not support license feature requested

Table 5-3: Errors related to the Compose process

String	Hex code	Decimal code	Description
CheckEventLog	0x8004f08a	-2147159926	General failure. See the event viewer for details.
profileForbidsPreview	0x8004f08f	-2147159921	This message cannot be sent. You do not have permission to request Fax Preview.
profileForbidsStatus	0x8004f090	-2147159920	This message cannot be sent. You do not have permission to request a status message.
profileMustSpecifyBillingCode	0x8004f091	-2147159919	This message cannot be sent without a billing code.
profileMustSpecifyAccessCode	0x8004f092	-2147159918	This message cannot be sent without an access code.
profileMustSpecifyCoverpage	0x8004f093	-2147159917	This message cannot be sent without a cover page.
profileMissing	0x8004f094	-2147159916	The profile assigned to you no longer exists. Until the problem with the profiles is resolved, you will not be able to send any messages.
SendOnBehalfNotAllowed	0x8004f098	-2147159912	You are not authorized to send on the behalf of the specified user.
OneOffNotAllowed	0x8004f099	-2147159911	Sending a Genifax without using the Genifax Form is not allowed.
invaldFormsOverlay	0x8004f09a	-2147159910	The specified forms overlay file is invalid.
Component_Lookup_Timeout			Lookup timeout.

Workflow errors

Table 5-4: **Errors related to general workflow**

String	Hex code	Decimal code	Description
PreviewRequestedNotConfigured	0x8004f384	-2147159164	Document preview was requested, but preview is not configured.
GlyphGeneralCodeNotFound	0x8004f385	-2147159163	Glyph General error: Error code not found.
GlyphParameterBadContext	0x8004f386	-2147159162	Glyph Parameter-related error; bad context.
GlyphParameterBadFormat	0x8004f387	-2147159161	Glyph Parameter-related error; bad format.
GlyphParameterBadParameter	0x8004f388	-2147159160	Glyph Parameter-related error; bad parameter.
GlyphMemoryNoMemoryAvailable	0x8004f389	-2147159159	Glyph Memory-related error; no memory.
GlyphFileOpenFailed	0x8004f38a	-2147159158	Glyph File-related error; file open failed.
GlyphFileReadFailed	0x8004f38b	-2147159157	Glyph File-related error; failed to read file the specified file.
GlyphFileWriteFailed	0x8004f38c	-2147159156	Glyph File-related error; failed to write to the specified file.
GlyphFileSeekFailed	0x8004f38d	-2147159155	Glyph File-related error; failed to seek to the correct position in specified file.
GlyphThreadNoAvailableThreads	0x8004f38e	-2147159154	Glyph Thread-related error; no threads available.
GlyphThreadFailedThread	0x8004f38f	-2147159153	Glyph Thread-related error; thread failed.
GlyphLogicalDecodeSynchronizationFailed	0x8004f390	-2147159152	Glyph Error related to logical decoding; decode synchronization failed.
GlyphLogicalDecodeMetaDecodeFailed	0x8004f391	-2147159151	Glyph Error related to logical decoding; meta decode failed.
GlyphLogicalDecodeDataDecodeFailed	0x8004f392	-2147159150	Glyph Error related to logical decoding; data decode failed.
GlyphLogicalDecodeKeyDecodeFailed	0x8004f393	-2147159149	Glyph Error related to logical decoding; key decode failed.
GlyphLogicalDecodeChecksumDecodeFailed	0x8004f394	-2147159148	Glyph Error related to logical decoding; checksum decode failed.

Table 5-4: Errors related to general workflow

String	Hex code	Decimal code	Description
GlyphLogicalDecodeInvalidBlockSize	0x8004f395	-2147159147	Glyph Error related to logical decoding; invalid block size.
GlyphIntializeFailedToInitGlyphSystem	0x8004f396	-2147159146	Glyph Error relating to toolkit initialization; failed to initialize system for glyph processing.
GlyphNonSpecific	0x8004f397	-2147159145	Glyph NonSpecific error.

Web Client errors

Table 5-5: **Errors related to the Web Client**

String	Hex code	Decimal code	Description
WebGeneralError	0x8004f3e8	-2147159064	A general Web error occurred.
WebFailedToFindXMLNode	0x8004f3e9	-2147159063	Could not find Specified XML Node.
WebParseHTMLForXMLError	0x8004f3f0	-2147159056	Error parsing the HTML for the XML structure.
WebInvalidFileHash	0x8004f3f1	-2147159055	Invalid File Hash.

DMS routing errors

Table 5-6: **Errors related to DMS routing**

String	Hex code	Decimal code	Description
DMSLogonFailed	0x8004f9c4	-2147157564	The session failed to log on to the DMS.
DMSUserCancelled	0x8004f9c5	-2147157563	The user cancelled the operation.
DMSSessionNotValid	0x8004f9c6	-2147157562	The DMS session is not active an/or valid.
DMSObjectNotFound	0x8004f9c7	-2147157561	The object was not found in the DMS database.
DMSMultipleUseNotSupported	0x8004f9c8	-2147157560	DMS provider does not support multiple use.
NoStubDocument			Stub document is not found.

DocCards

Table 5-7: **Errors related to DocCards**

String	Hex code	Decimal code	Description
NoSigningCertificate	0x8004f7d0	-2147158064	No signing certificate
MsgPropertyRetrievalFailure	0x8004f7d1	-2147158063	Message missing required information

Table 5-7: **Errors related to DocCards**

String	Hex code	Decimal code	Description
XMLInformationRetrievalFailure	0x8004f7d2	-2147158062	Failed to retrieve XML information required for translation
SigningCertificateFailure	0x8004f7d3	-2147158061	Missing or invalid certificate
GenidocExpired	0x8004f7d4	-2147158060	Expiration date has already been exceeded

Warnings for successful functions

Table 5-8: **Warnings**

String	Hex	Decimal	Description
Warning_NoDataFound	0x4e1f4	319988	Function succeeded. No data found.
Warning_DataNotAvaliable	0x4e1f5	319989	Function succeeded. There is data but it cannot be accessed yet. Try again later.
Warning_DeliverPending	0x4e1f6	319990	Deliver is pending; move to "deliver pending".
Warning_CancelPending	0x4e1f7	319991	Cancellation is pending.

Section 6: Examples of Common Functions

This section includes:

Adding a file attachment to a message

Remarks

A message must have at least one recipient and one file attachment.

The message attachment object has a required property, `AttachmentOriginalPathName`, and an optional property, `IsEmailBody`.

Example

'Create a new message object:

```
Dim oNewMessage As New Message
Set oNewMessage = oMsgContainer.New
```

'Create a message attachment object:

```
Dim oNewAttachment As New MessageAttachment
```

'Set the required property on the message attachment object:

```
oNewAttachment.AttachmentOriginalPathName = [bstr value]
```

```
'Set an optional property on the message attachment object:
oNewAttachment.IsEmailBody = [bool value]

'Add the message attachment object to the message object:
oNewMessage.Add oNewAttachment
```

Adding a file attachment to an Embedded Directive

Remarks

An Embedded Directive must have at least one recipient. File attachments are optional.

The required property on the attachment object is `AttachmentOriginalPathName`. The optional property that can be set on the attachment object is `IsEmailBody`.

Example

```
'Create an Embedded Directive object:
Dim oNewED As Object
Set oNewED = oEDContainer.New

'Create a message attachment object:
Dim oNewAttachment As New MessageAttachment

'Set the required property on the message attachment object:
oNewAttachment.AttachmentOriginalPathName = [bstr value]

'Set an optional property on the message attachment object:
oNewAttachment.IsEmailBody = [bool value]

'Add the message attachment object to the Embedded Directive object:
oNewED.Add oNewAttachment
```

Adding a recipient to a message

Remarks

A message must have at least one recipient and one file attachment.

The message recipient object has required properties, `Destination` and `RecipientType`, and numerous optional properties.

Example

```
'Create a new message object:
Dim oNewMessage As New Message
Set oNewMessage = oMsgContainer.New
```

```
'Create the recipient object:
Dim oMessageRecipient As New MessageRecipient

'Set the required properties on the recipient object:
oMessageRecipient.Destination = [bstr value]
oMessageRecipient.RecipientType = [ApiRecipientType]

'Set some optional properties on the recipient object:
oMessageRecipient.AccessCode = [bstr value]
oMessageRecipient.ApprovalRequested = [bool value]
oMessageRecipient.BillingCode = [bstr value]
oMessageRecipient.DateSendAfter = [date value]
oMessageRecipient.FinalFormCode = [bstr value]
oMessageRecipient.Notification = [NotificationType]
oMessageRecipient.PreviewRequired = [bool value]
oMessageRecipient.PrintedByWebSite = [bool value]
oMessageRecipient.Priority = [PriorityType]
oMessageRecipient.SecureFax = [bool value]
oMessageRecipient.TemplateFilename = [bstr value]
oMessageRecipient.ViewedByWebSite = [bool value]

'Add the recipient object to the message object:
oNewMessage.Add oMessageRecipient
```

Adding a recipient to an Embedded Directive

Remarks

An Embedded Directive must have at least one recipient. File attachments are optional.

The message recipient object has required properties, Destination and RecipientType, and numerous optional properties.

Example

```
'Create an Embedded Directive object:
Dim oNewED As Object
Set oNewED = oEDContainer.New

'Create the recipient object:
Dim oEDRecipient As New MessageRecipient

'Set the required properties on the recipient object:
oEDRecipient.Destination = [bstr value]
oEDRecipient.RecipientType = [ApiRecipientType]

'Set some optional properties on the recipient object:
oEDRecipient.AccessCode = [bstr value]
oEDRecipient.ApprovalRequested = [bool value]
oEDRecipient.BillingCode = [bstr value]
oEDRecipient.DateSendAfter = [date value]
oEDRecipient.FinalFormCode = [bstr value]
oEDRecipient.Notification = [NotificationType]
```

```
oEDRecipient.PreviewRequired = [bool value]
oEDRecipient.PrintedByWebSite = [bool value]
oEDRecipient.Priority = [PriorityType]
oEDRecipient.SecureFax = [bool value]
oEDRecipient.TemplateFilename = [bstr value]
oEDRecipient.ViewedByWebSite = [bool value]
```

'Add the recipient object to the Embedded Directive object:

```
oNewED.Add oEDRecipient
```

Connecting to the server

Remarks

You can connect to the message server as a user or as an administrator.

When you connect to the message server as a user and get a collection, it contains only items associated with the user. Additionally, any objects you create, such as messages and Embedded Directives, become associated with the user. Connect to the message server as a user when the functions in the application or script are user-specific, for example, when creating messages and Embedded Directives.

When you connect to the message server as an administrator and get a collection, it contains items associated with all users. Connect to the message server as an administrator when the functions in the application or script are not user-specific.

Example: Administrator

'Create a server connection object:

```
Dim oServConnect As Object
Set oServConnect = CreateObject("Omtool.ServConnect.1")
```

'Check the creation:

```
If (oServConnect Is Nothing) Then
GoTo ErrorHandler
End If
```

'Connect to the message server:

```
Dim oMsgServer As Object
Set oMsgServer = oServConnect.ConnectToServerAdmin("servername")
```

'Check the connection:

```
If (oMsgServer Is Nothing) Then
GoTo ErrorHandler
End If
```

Example: User

'Create a server connection object:

```
Dim oServConnect As Object
Set oServConnect = CreateObject("Omtool.ServConnect.1")
```

'Check the creation:
```
If (oServConnect Is Nothing) Then
GoTo ErrorHandler
End If
```

'Connect to the message server:
```
Dim oMsgServer As Object
Set oMsgServer = oServConnect.ConnectToServer("servername", "username")
```

'Check the connection:
```
If (oMsgServer Is Nothing) Then
GoTo ErrorHandler
End If
```

Creating a message with a recipient and file attachment

Remarks

A message must have at least one recipient and one file attachment.

The message recipient object has required properties, Destination and RecipientType, and numerous optional properties.

The message attachment object has a required property, AttachmentOriginalPathName, and an optional property, IsEmailBody.

Example

'Create a new message object:
```
Dim oNewMessage As New Message
Set oNewMessage = oMsgContainer.New
```

'Create a recipient object and add it to the message object:
```
Dim oMessageRecipient As New MessageRecipient
oMessageRecipient.Destination = [bstr value]
oMessageRecipient.RecipientType = [ApiRecipientType]
oNewMessage.Add oMessageRecipient
```

'Create a message attachment object and add it to the message object:
```
Dim oNewAttachment As New MessageAttachment
oNewAttachment.AttachmentOriginalPathName = [bstr value]
oNewMessage.Add oNewAttachment
```

'Save the message object and submit it to the message server:
```
oNewMessage.Save
```

Creating a message with a Routing Sheet attachment

Remarks

A message must have at least one recipient and one file attachment.

The message recipient object has required properties, Destination and RecipientType, and numerous optional properties.

The message attachment object has a required property, AttachmentOriginalPathName, and an optional property, IsEmailBody.

Example

```
'Create an Embedded Directive object:
Dim oNewED As Object
Set oNewED = oEDContainer.New

'Set some optional properties on the Embedded Directive object:
oNewED.SingleUse = [bool value]
oNewED.Title = [bstr value]
oNewED.ApplicationName = [bstr value]
oNewED.ApplicationTag = [bstr value]

'Create a recipient object:
Dim oEDRecipient As New MessageRecipient

'Set the required properties on the recipient object:
oEDRecipient.Destination = [bstr value]
oEDRecipient.RecipientType = [ApiRecipientType]

'Set some optional properties on the recipient object:
oEDRecipient.AccessCode = [bstr value]
oEDRecipient.ApprovalRequested = [bool value]
oEDRecipient.BillingCode = [bstr value]
oEDRecipient.DateSendAfter = [date value]
oEDRecipient.FinalFormCode = [bstr value]
oEDRecipient.Inbound = [bool value]
oEDRecipient.Notification = [NotificationType]
oEDRecipient.PreviewRequired = [bool value]
oEDRecipient.PrintedByWebSite = [bool value]
oEDRecipient.Priority = [PriorityType]
oEDRecipient.SecureFax = [bool value]
oEDRecipient.Subject = [bstr value]
oEDRecipient.TemplateFilename = [bstr value]
oEDRecipient.ViewedByWebSite = [bool value]

'Add the recipient to the Embedded Directive object:
oNewED.Add oEDRecipient

'Create a Routing Sheet template object:
Dim oEmbeddedDirectivesTemplates As Object
Dim oEDTemplate As Object
Set oEmbeddedDirectivesTemplates = oMsgServer.EmbeddedDirectivesTemplates
Set oEDTemplate = oEmbeddedDirectivesTemplates(1)
Set oEmbeddedDirectivesTemplates = Nothing
oNewED.StyleEntryID = oEDTemplate.EntryID

'Save the Embedded Directive object:
oNewED.Save
```

```
'Create the Routing Sheet:
Dim CurrentPath As String
Dim RoutingSheetPath As String
CurrentPath = App.Path
RoutingSheetPath = oNewED.ComposeRoutingSheet(CurrentPath)

'Clear the oEDRecipient object:
Set oEDRecipient = Nothing

'Create a new message object:
Dim oNewMessage As New Message
Set oNewMessage = oMsgContainer.New

'Add the Embedded Directive object to the message object:
Dim oMessageRecipient As New MessageRecipient
oMessageRecipient.Destination = oNewED.ID
oMessageRecipient.RecipientType = EmbeddedDirective
oNewMessage.Add oMessageRecipient
```

Note that this associates the Embedded Directive with the message, and the next section of code attaches the Routing Sheet. (A message requires both a recipient object and a message attachment object.)

When attaching a Routing Sheet, you are not required to also associate an Embedded Directive with the message; however, the message must have a recipient object. As an alternative, you can specify a recipient and destination address that matches on a rule so that the message is routed to the Embedded Directive Manager component, and/or specify an additional recipient of the message.

```
'Attach the Routing Sheet to the message object:
Dim oNewAttachment As New MessageAttachment
oNewAttachment.AttachmentOriginalPathName = RoutingSheetPath
oNewMessage.Add oNewAttachment

'Save the message object and submit it to the message server:
oNewMessage.Save
```

Creating a message with an additional notification recipient

Remarks

A message must have at least one recipient and one file attachment.

The message recipient object has required properties, Destination and RecipientType, and numerous optional properties.

The message attachment object has a required property, AttachmentOriginalPathName, and an optional property, IsEmailBody.

In this example, the notification type NotifyOnBoth is set on the message object, ensuring that the originator of the message receives a notification message when the message server delivers or fails the message. Then a notification recipient is added. Because the property Notification is not set on the recipient object, the notification recipient inherits the notification setting, NotifyOnBoth, from the message object.

Example

'Create a new message object:
```
Dim oNewMessage As New Message
Set oNewMessage = oMsgContainer.New
```

'Set the notification type for the message object:
```
oNewMessage.Notification = NotifyOnBoth
```

'Create a notification recipient object and add it to the message object:
```
Dim oNotifRecipient As New NotificationRecipient
oNotifRecipient.Destination = [bstr value]
oNotifRecipient.RecipientType = [ApiRecipientType]
oNewMessage.Add oNotifRecipient
```

'Create a recipient object and add it to the message object:
```
Dim oMessageRecipient As New MessageRecipient
oMessageRecipient.Destination = [bstr value]
oMessageRecipient.RecipientType = [ApiRecipientType]
oNewMessage.Add oMessageRecipient
```

'Create a message attachment object and add it to the message object:
```
Dim oNewAttachment As New MessageAttachment
oNewAttachment.AttachmentOriginalPathName = [bstr value]
oNewMessage.Add oNewAttachment
```

'Save the message object and submit it to the message server:
```
oNewMessage.Save
```

Creating a message with an Embedded Directive

Remarks

A message must have at least one recipient and one file attachment.

The message recipient object has required properties, Destination and RecipientType, and numerous optional properties.

The message attachment object has a required property, AttachmentOriginalPathName, and an optional property, IsEmailBody.

Example

'Create a new message object:
```
Dim oNewMessage As New Message
Set oNewMessage = oMsgContainer.New
```

'Add the Embedded Directive object to the message object:
```
Dim oMessageRecipient As New MessageRecipient
oMessageRecipient.Destination = oED.ID
oMessageRecipient.RecipientType = EmbeddedDirective
oNewMessage.Add oMessageRecipient
```

'Create a message attachment object and add it to the message object:
```
Dim oNewAttachment As New MessageAttachment
oNewAttachment.AttachmentOriginalPathName = [bstr value]
oNewMessage.Add oNewAttachment
```

'Save the message object and submit it to the message server:
```
oNewMessage.Save
```

Creating a Routing Sheet

Remarks

You can create a Routing Sheet from an existing Embedded Directive or a new Embedded Directive.

Example

'Create an Embedded Directive object:
```
Dim oNewED As Object
Set oNewED = oEDContainer.New
```

'Set some optional properties on the Embedded Directive object:
```
oNewED.SingleUse = [bool value]
oNewED.Title = [bstr value]
oNewED.ApplicationName = [bstr value]
oNewED.ApplicationTag = [bstr value]
```

'Create a recipient object and add it to the Embedded Directive object:
```
Dim oEDRecipient As New MessageRecipient
oEDRecipient.Destination = [bstr value]
oEDRecipient.RecipientType = [ApiRecipientType]
oEDRecipient.Priority = [PriorityType]
oNewED.Add oEDRecipient
```

'Create a Routing Sheet template object:
```
Dim oEmbeddedDirectivesTemplates As Object
Dim oEDTemplate As Object
Set oEmbeddedDirectivesTemplates = oMsgServer.EmbeddedDirectivesTemplates
Set oEDTemplate = oEmbeddedDirectivesTemplates(1)
Set oEmbeddedDirectivesTemplates = Nothing
oNewED.StyleEntryID = oEDTemplate.EntryID
```

'Save the Embedded Directive object:
```
oNewED.Save
```

'Create Routing Sheet:
```
Dim CurrentPath As String
Dim RoutingSheetPath As String
CurrentPath = App.Path
RoutingSheetPath = oNewED.ComposeRoutingSheet(CurrentPath)
```

Creating an Embedded Directive

Remarks

An Embedded Directive must have at least one recipient.

The message recipient object has required properties, Destination and RecipientType, and numerous optional properties.

There are no required properties on an Embedded Directive object, but numerous option properties.

Example

'Create an Embedded Directive object:
```
Dim oNewED As Object
Set oNewED = oEDContainer.New
```

'Set some optional properties on the Embedded Directive object:
```
oNewED.SingleUse = [bool value]
oNewED.Title = [bstr value]
oNewED.ApplicationName = [bstr value]
oNewED.ApplicationTag = [bstr value]
```

'Create a recipient object:
```
Dim oEDRecipient As New MessageRecipient
```

'Set the required properties on the recipient object:
```
oEDRecipient.Destination = [bstr value]
oEDRecipient.RecipientType = [ApiRecipientType]
```

'Set some optional properties on the recipient object:
```
oEDRecipient.AccessCode = [bstr value]
oEDRecipient.ApprovalRequested = [bool value]
oEDRecipient.BillingCode = [bstr value]
oEDRecipient.DateSendAfter = [date value]
oEDRecipient.FinalFormCode = [bstr value]
oEDRecipient.Notification = [NotificationType]
oEDRecipient.PreviewRequired = [bool value]
oEDRecipient.PrintedByWebSite = [bool value]
oEDRecipient.Priority = [PriorityType]
oEDRecipient.SecureFax = [bool value]
oEDRecipient.TemplateFilename = [bstr value]
oEDRecipient.ViewedByWebSite = [bool value]
```

'Add the recipient object to the Embedded Directive object:
```
oNewED.Add oEDRecipient
```

'Save the Embedded Directive object:
```
oNewED.Save
```

Setting template variables on a message

Remarks

A complete list of supported template variables, `CoverPageVar.RTF`, resides on the message server in `...\Omtool\OmtoolServer\Languages\xxx\Templates`.

When citing the name of a template variable, remember to omit the percent symbols that would normally be included when inserting the variable into a template file.

Note that you can set template variables on message objects and recipient objects.

This example illustrates template variables set on the message object.

Example

```
'Create a new message object:
Dim oNewMessage As New Message
Set oNewMessage = oMsgContainer.New

'Create a recipient object and add it to the message object:
Dim oMessageRecipient As New MessageRecipient
oMessageRecipient.Destination = [bstr value]
oMessageRecipient.RecipientType = [ApiRecipientType]
oNewMessage.Add oMessageRecipient

'Create a message attachment object and add it to the message object:
Dim oNewAttachment As New MessageAttachment
oNewAttachment.AttachmentOriginalPathName = [bstr value]
oNewMessage.Add oNewAttachment

'Set template variables on the message object:
'For example: oNewMessage.SetTemplateVar "SENDER_NAME", "Claims handler"
oNewMessage.SetTemplateVar [bstrName], [bstrVal]
oNewMessage.SetTemplateVar [bstrName], [bstrVal]
oNewMessage.SetTemplateVar [bstrName], [bstrVal]
oNewMessage.SetTemplateVar [bstrName], [bstrVal]
oNewMessage.SetTemplateVar [bstrName], [bstrVal]

'Save the message object and submit it to the message server:
oNewMessage.Save
```

Setting template variables on a recipient

Remarks

A complete list of supported template variables, CoverPageVar.RTF, resides on the message server in `...\Omtool\OmtoolServer\Languages\xxx\Templates`.

When citing the name of a template variable, remember to omit the percent symbols that would normally be included when inserting the variable into a template file.

Note that you can set template variables on message objects and recipient objects.

This example illustrates template variables set on the recipient object.

Example

```
'Create the recipient object:
Dim oMessageRecipient As New MessageRecipient

'Set the required properties on the recipient object:
oMessageRecipient.Destination = [bstr value]
oMessageRecipient.RecipientType = [ApiRecipientType]

'Set some optional properties on the recipient object:
oMessageRecipient.AccessCode = [bstr value]
oMessageRecipient.ApprovalRequested = [bool value]
oMessageRecipient.BillingCode = [bstr value]
oMessageRecipient.DateSendAfter = [date value]
oMessageRecipient.FinalFormCode = [bstr value]
oMessageRecipient.Notification = [NotificationType]
oMessageRecipient.PreviewRequired = [bool value]
oMessageRecipient.PrintedByWebSite = [bool value]
oMessageRecipient.Priority = [PriorityType]
oMessageRecipient.SecureFax = [bool value]
oMessageRecipient.TemplateFilename = [bstr value]
oMessageRecipient.ViewedByWebSite = [bool value]

'Set template variables on the recipient object:
'For example: oMessageRecipient.SetTemplateVar "APPROVAL_MANAGER", "Manager, Claims Processing"
oMessageRecipient.SetTemplateVar [bstrName], [bstrVal]
oMessageRecipient.SetTemplateVar [bstrName], [bstrVal]
oMessageRecipient.SetTemplateVar [bstrName], [bstrVal]
oMessageRecipient.SetTemplateVar [bstrName], [bstrVal]
oMessageRecipient.SetTemplateVar [bstrName], [bstrVal]
```