ExEMail

A built from the ground up using 100% ATL-based code, The ExEMail can be dropped into any language that supports ActiveX thus enabling an application to support full mail and file attachment transfers. ExEmail is fully SMTP RFC-compliant and provides support for binary (MIME) attachments and HTML email.

What does ExEMail provide and other SMTP components doesn't?

The ExEMail is able to find the SMTP server where the email message should be sent, from the recipient's email address, by queering a DNS server. The ExEMail doesn't block your application while delivering a message. It provides a range of events to let you know how the email message is delivered. The object model is intuitive, rich and flexible.

How ExEMail component delivers a message?

Here are the steps that ExEMail component follows in order to deliver an email message:

1. The component sends a query to the DNS server in order to get the SMTP hosts that are responsible for the domain recipient's email address.
2. Based on the DNS's response, the component is trying each SMTP server found to deliver the email message.
3. The component prepares and sends the information about the sender and recipients email addresses to the SMTP server.
4. If the server accepts the sender and the recipients, it prepares the message's data and sends it.
5. Once that message's data was sent, the component is going to close the connection.
6. The client is going to inform the server that wants to close the connection, by sending QUIT command
7. The client is disconnected

Here's a sample how to send an email message using non-blocking mode:

```
Dim WithEvents e As EMail

Private Sub e_EndSend(ByVal Msg As EXEMAILLibCtl.IMessage)
    If Msg.LastError = 0 Then
        MsgBox "The message was succesfully delivered."
    End If
End Sub
```

```
Private Sub Form_Load()
    Set e = New EMail
    Dim m As New Message
    Set m.Notifier = e

    m.Send "myaccount@usermail.com", "sss@colam.com", "Test", "This is a test message",
"c:\winnt\system32\setup.exe"
End Sub
```

The non-blocking mode does not block your application during delivering the messages, and your application can handle each message' notifications (events).

The following sample shows how to send an email message using blocking mode:

```
Private Sub Form_Load()
    Dim m As New Message
    If (0 = m.Send("myaccount@usermail.com", "sss@colam.com", "Test", "This is a test
message", "c:\winnt\system32\setup.exe")) Then
        MsgBox "The message was succesfully delivered "
    End If
End Sub
```

The blocking mode blocks your application during delivering the message, and it doesn't fire any event. In order to create EMail and Message object for your application you have to use NewEmail and NewMessage properties that manages to license the objects at runtime.

How to get support?

To keep your business applications running, you need support you can count on.

Here are few hints what to do when you're stuck on your programming:

- Check out the samples - they are here to provide some quick info on how things should be done
- Check out the how-to questions using the eXHelper tool
- Check out the help - includes documentation for each method, property or event
- Check out if you have the latest version, and if you don't have it send an update request here.
- Submit your problem(question) here.

Don't forget that you can contact our development team if you have ideas or requests for new components, by sending us an e-mail at support@exontrol.com (please include the name of the product in the subject, ex: exgrid) . We're sure our team of developers will try to find a way to make you happy - and us too, since we helped.

Regards,
Exontrol Development Team

https://www.exontrol.com

constants AuthMethodEnum

Specifies the authentication method.

Name	Value	Description
NoAuth	0	No Authentification.
AuthLogin	1	Login
AuthPlain	2	Plain

constants BodyEncodingEnum

Specifies the type of encoding the message' body supports. The BodyEncoding property specifies the way the body of the message is encoded. By default, the message's body goes as BASE64. The BodyEncodingEnum type supports the following values:

Name	Value	Description
NoBodyEncoding	0	The message's body encoding could be 7BIT or 8BIT.
BodyEncodingBASE64	1	The message's body encoding is BASE64 (by default).

constants PriorityEnum

Specifies the email message's priority.

Name	Value	Description
High	1	High priority.
Normal	3	Normal priority (Default).
Low	5	Low priority.

Use the Priority property to change the message's priority.

constants SendStateEnum

Specifies the state of the message while it is delivering.

Name	Value	Description
LookupMX	1	The component is looking for mail exchange information, by queering the DNS server.
Connecting	2	The client is going to connect to the SMTP server.
Connected	3	The client is connected to the SMTP server.
Opening	4	The connection is going to be opened, by saying HELO to the SMTP server.
Opened	5	The connection is opened, and ready to send the email.
Closing	6	The connection is about to be closed.
Closed	7	The connection is closed.
Data	8	The client sends the message's data.
Disconnecting	9	The client is going to be disconnected from the SMTP server
Disconnected	10	The client is disconnected.
Login	11	The client is about to login to the SMTP server. The Login state is sent only if the AuthMethod property is AuthLogin or AuthPlain.
Logged	12	The client is logged to the SMTP server. The Logged state is sent only if the AuthMethod property is AuthLogin or AuthPlain.

Use the StateChange event to check the email message's state while delivering.

EMail object

The EMail object provides a range of events that let the user know how the message delivering goes. In order to create a licensed EMail object, you have to use NewEmail property. If you deploy an application that uses CreateObject or New statements in order to create new EMail objects, it fails because the EMail object is a licensed ActiveX component. The New and CreateObject statements work only on machines where the ExEMail component was licensed for design mode. A Message object sent notifications (events) to the application only if Notifier property points to an EMail object. Use the EMail object to handle message' notifications, or for retrieving a description for the last error occured. Use the NewMessage to create licensed Message objects at runtime.

For instance, the following sample shows how to send an email message using non-blocking mode:

```
Dim WithEvents e As EMail

Private Sub e_EndSend(ByVal Msg As EXEMAILLibCtl.IMessage)
    Debug.Print e.Error(Msg.LastError)
End Sub

Private Sub Form_Load()
    ' Creates an EMail object, that will receive Message notifications
    Set e = Runtime1.NewEMail()
    ' Creates a Message object
    Dim m As Message
    Set m = Runtime1.NewMessage()
    ' Message' notifications are sent through EMail object
    Set m.Notifier = e
    ' Sends a message to one user
    m.Send "m@malibu.com", "marfa@cool2.com", "Test", "The message has one 
attchament", "c:\winnt\system32\setup.exe"
End Sub
```

The following sample shows how to send an email message using blocking mode:

```
Private Sub Form_Load()
    Dim e As EMail
    ' Creates an EMail object, that will receive Message notifications
    Set e = Runtime1.NewEMail()
    ' Creates a Message object
    Dim m As Message
    Set m = Runtime1.NewMessage()
    ' Sends a message to one user
    Debug.Print e.Error(m.Send("m@malibu.com", "marfa@cool2.com", "Test", "The
message has one attchament", "c:\winnt\system32\setup.exe"))
End Sub
```

The samples do the same thing, but the difference between them is that non-blocking mode fires events, and doesn't block your application, and the blocking mode doesn't fire events, and it blocks your application during delivering the message.

Name	Description
error	Gets description for an error code.

property EMail.error (Code as Long) as String

Gets description for an error code.

Type	Description
Code as Long	A long expression that describes the error code
String	A string expression that describes the error.

Use the Error property to get a description for an error code. Use the LastError property to get the last error occurred in delivering the message. If the Notifier property points to an EMail object, use the EndSend event to check the last error occured. The following sample shows how to check whether a message was successfully delivered, using non-blocking mode:

```
Dim WithEvents e As EMail

Private Sub e_EndSend(ByVal Msg As EXEMAILLibCtl.IMessage)
    If (Msg.LastError <> 0) Then
        Debug.Print e.Error(Msg.LastError)
    End If
End Sub

Private Sub Form_Load()

    ' Creates an EMail object, that will receive Message notifications
    Set e = Runtime1.NewEMail()
    ' Creates a Message object
    Dim m As Message
    Set m = Runtime1.NewMessage()
    ' Message' notifications are sent through EMail object
    Set m.Notifier = e

    ' Sends a message to one user
    m.Send "m@malibu.com", "marfa@cool2.com", "Test", "The message has one
attchament", "c:\winnt\system32\setup.exe"
End Sub
```

The following sample shows how to print the description for last error occurred using blocking mode (when the Notifier property is set to nothing). It's not recommended using blocking mode because it blocks your application while the message is delivered.

```
MsgBox Runtime1.NewEMail.Error(Runtime1.NewMessage().Send("m@malibu.com",
"marfa@cool2.com", "Test", "The message has one attchament",
"c:\winnt\system32\setup.exe"))
```

The following sample shows how to send an email message using a single line of code:

```
If 0 = Runtime1.NewMessage().Send("m@malibu.com", "marfa@cool2.com", "Test", "The
message has one attchament", "c:\winnt\system32\setup.exe") Then
    MsgBox "OK"
End If
```

Message object

A Message object holds information about an email message. Use the Send method to send the email message. Use the NewMessage property to create a licensed Message object. Using the ExEMail component there are two ways to send email messages: non-blocking and blocking mode. The following sample shows how to send an email message using non=blocking mode:

```
Dim WithEvents e As EMail

Private Sub Form_Load()
    ' Creates an EMail object, that will receive Message notifications
    Set e = Runtime1.NewEMail()
    ' Creates a Message object
    Dim m As Message
    Set m = Runtime1.NewMessage()
    ' Message' notifications are sent through EMail object
    Set m.Notifier = e
    ' Sends a message to 2 users
    m.Send "me@malibu.com", "bula@bulat.com,robingo@calcuta.fom", "Test", "The
message has one attchament", "c:\winnt\system32\setup.exe"
End Sub
```

The following sample sends the email message by blocking the application while it delivers the message:

```
Private Sub Form_Load()
    ' Creates a Message object
    Dim m As Message
    Set m = Runtime1.NewMessage()
    ' Sends a message to 2 users
    m.Send "me@malibu.com", "bula@bulat.com,robingo@calcuta.fom", "Test", "The
message has one attchament", "c:\winnt\system32\setup.exe"
End Sub
```

Here's the list of supported properties:

Name	Description
	Retrieves or sets a value that indicates a list of files

Attachment	attached to the message separated by semicolon.
AuthMethod	Retrieves or sets a value that indicates the authentication method.
Bcc	Retrieves or sets a value that indicates the blind carbon copy recipient's email address(s) (separated by commas, if there are more than once).
BodyEncoding	Specifies the way the message's body is encoded.
BodyHTML	Retrieves or sets a the message's body in HTML format. The HTML content is attached as an alternative to plain text.
BodyText	Retrieves or sets the message's body as plain text.
Cc	Retrieves or sets a value that indicates the carbon copy recipient's email address(s) (separated by commas, if there are more than once).
Date	Retrieves or sets the message's date.
DNS	Retrieves the list of DNS servers.
ExtraHeader	Retrieves or sets any extra field to the message's header.
From	Retrieves or sets a value that indicates the email address of the sender.
Helo	Specifies the string being sent by HELO command to the e-mail server. As a result, the receiver-SMTP will not have to perform MX resolution on this name in order to validate the HELO parameter.
Host	Retrieves or sets a value that indicates the host where the message should be delivered. If the Host is empty, the control queries the DNS where the message should be delivered.
LastError	Retrieves the last error that occurs in delivering the message.
LogonPassword	Retrieves or sets the value that indicates the login's password.
LogonUser	Retrieves or sets a value that indicates the login user name.
Notifier	Changes or gets the message's object notifier. The object notifier receives all the message events.
Port	Retrieves or sets the server's port.

Priority	Retrieves or sets a value that indicates the message's priority.
Send	Sends the email message to the recipients.
SendOnce	Indicates whether the message is sent once to the mail server, for all recipients, and the server takes the responsibility to delivery the message per all assigned recipients.
Subject	Retrieves or sets a value that indicates the subject of the message.
Timeout	Specifies the amount of time (in seconds) the control will wait for the server response.
To	Retrieves or sets a value that indicates the recipient's email address, or a list of recipient's email addresses separated by commas.

property Message.Attachment as String

Retrieves or sets a value that indicates the list of message' attached files separated by semicolon.

Type	Description
String	A String expression that indicates the files to attach as: • a list of files (including the full path) to attach, separated by semicolon • a multi-line string defining the file name to attach, with subsequent lines specifying the content of the file to attach (ability to attach a file with its content without physically having it)

To include attachments in your email message, there are two methods you can use:

- Attachment property: Set the Attachment property of your email message object. This property allows you to specify the files you want to attach directly within your email composition process
- Attachment argument of Send method: Alternatively, when calling the Send method to send your email, you can pass an Attachment argument. This argument lets you include files as attachments at the time of sending the email

Both approaches provide flexibility in how you attach files to your emails, depending on your specific programming or email sending scenario.

The following method shows two different ways how you can attach files to a Message object:

```
Private Sub Form_Load()
   Dim m As Message
   Set m = Runtime1.NewMessage
   If (0 = m.Send("myaccount@usermail.com", "sss@colam.com", "Test", "This is a test message", "c:\winnt\system32\setup.exe;c:\winnt\system32\setup.bin")) Then
      MsgBox "The message was succesfully delivered "
   End If
End Sub
```

```
Private Sub Form_Load()
   Dim m As Message
```

```
    Set m = Runtime1.NewMessage
    m.Attachment = "c:\winnt\system32\setup.exe;c:\winnt\system32\setup.bin"
    If (0 = m.Send("myaccount@usermail.com", "sss@colam.com", "Test", "This is a test
message")) Then
        MsgBox "The message was succesfully delivered "
    End If
End Sub
```

property Message.AuthMethod as AuthMethodEnum

Retrieves or sets a value that indicates the authentication method.

Type	Description
AuthMethodEnum	An AuthMethodEnum expression that indicates the authentication method.

By default, the control uses no authentication method. Use the AuthMethod property to specifies the authentication method. The "AUTH LOGIN" and "AUTH PLAIN" supported. The authentication method depends on the SMTP server you are using. Use the LogonUser and LogonPassword properties if the authentication method is AuthLogin or AuthPlain. The RFC 2554 describes the SMTP Authentication.

property Message.Bcc as String

Retrieves or sets a value that indicates the blind carbon copy recipient's email address's) separated by commas.

Type	Description
String	A string expression that indicates the blind carbon copy recipient's email address's) separated by commas.

Use the Bcc property to add blind carbon copy email addresses to your email message. By default, the Bcc property is empty. If the Bcc is empty, no Bcc field is added to message's header.

The following sample shows how to add Cc and Bcc fields to your message:

```
Dim m As Message
Set m = Runtime1.NewMessage
m.Cc = "norton@macrosoft.com,steve@segal.com"
m.Bcc = """Bill Gates"" <bil@macrosoft.com>,bala@segal.com"
If (0 = m.Send("myaccount@usermail.com", "sss@colam.com", "Test", "This is a test
message")) Then
    MsgBox "The message was successfully delivered. "
End If
```

property Message.BodyEncoding as BodyEncodingEnum

Specifies the way the message's body is encoded.

Type	Description
BodyEncodingEnum	A BodyEncodingEnum expression that defines the encoding type of the message's body.

By default, the BodyEncoding property is BodyEncodingBASE64. Use the BodyEncoding property to specify the encoding type of the message's body. The BodyText / BodyHTML property specifies the message's body to be sent.

The BodyEncoding property could be:

- NoBodyEncoding, which indicates that the message's body goes as 7BIT or 8BIT
- BodyEncodingBASE64, which indicates that the message's body is encoded as BASE64.

property Message.BodyHTML as String

Retrieves or sets a the message's body in HTML format. The HTML content is attached as an alternative to BodyText text.

Type	Description
String	A String expression that indicates the HTML alternative for the BodyText property.

Use the BodyHTML property to send your message in HTML format. Use the BodyEncoding property to specify the encoding type of the message's body. If the property is empty, the email message is sent as plain text.

Here's a sample that shows you how to send an HTML message:

```
Dim m As Message
Set m = Runtime1.NewMessage
m.BodyHTML = "<html>Hello world</html>"
If (0 = m.Send("myaccount@usermail.com", "sss@colam.com", "HTML format", "Hello world")) Then
   MsgBox "The message was successfully delivered "
End If
```

property Message.BodyText as String

Retrieves or sets the message's body as plain text.

Type	Description
String	A String expression that indicates the message's body.

Use the BodyText to set the message's body. Use the BodyHTML to send your email message in HTML format. You can set also the email message's body by setting the Message optional parameter of the Send method. Use the BodyEncoding property to specify the encoding type of the message's body.

The following samples show how to set the message's text:

```
Dim m As Message
Set m = Runtime1.NewMessage
m.BodyText = "Hello world!"
If (0 = m.Send("myaccount@usermail.com", "bul@bulstone.com", "Test")) Then
   MsgBox "The message was successfully delivered "
End If
Dim m As Message
Set m = Runtime1.NewMessage
If (0 = m.Send("myaccount@usermail.com", "bul@bulstone.com", "Test", "Hello world!"))
Then
   MsgBox "The message was successfully delivered "
End If
```

property Message.Cc as String

Retrieves or sets a value that indicates the carbon copy recipient's email address's separated by commas

Type	Description
String	A String expression that indicates the carbon copy recipient's email address's separated by commas

Use Cc property to set Cc (Carbon Copy) field for your email message. By default, the Cc property is empty. If the Cc property is empty, no Cc field is added to message's header.

The following sample shows how to add Cc and Bcc fields to your message:

```
Dim m As Message
Set m = Runtime1.NewMessage
m.Cc = "norton@macrosoft.com,steve@segal.com"
m.Bcc = """Bill Gates"" <bil@macrosoft.com>,bala@segal.com"
If (0 = m.Send("myaccount@usermail.com", "sss@colam.com", "Test", "This is a test
message")) Then
   MsgBox "The message was successfully delivered "
End If
```

property Message.Date as Date

Retrieves or sets the message's date.

Type	Description
Date	A Date expression that indicates the message's date.

Use the Date property to set the message's date. By default the message's date is the current date.

The following sample shows how to send an message dated as yesterday:

```
Dim m As Message
Set m = Runtime1.NewMessage
m.Date = Date - 1
If (0 = m.Send("myaccount@usermail.com", "bul@bulstone.com", "Test", "Hello world!"))
Then
   MsgBox "The message was successfully delivered "
End If
```

property Message.DNS as String

Retrieves the list of DNS servers.

Type	Description
String	A string expression that indicates the list of local DNS servers.

The DNS property gets the list of the local DNS servers separated by comma. The control sends queries to a DNS server to determine the SMTP server that's responsible for an e-mail address.

property Message.ExtraHeader(Field as String) as String

Retrieves or sets any extra field to the message's header.

Type	Description
Field as String	A string expression that indicates the field's name.
String	A string expression that indicates the field's value.

Use the ExtraHeader property to add any extra field to the message's header. You can use your personalize fields, or you can check the RFC 822. The following sample shows how to add the "Reply-To" field to the message's body:

```
Private Sub Form_Load()
    Dim m As Message
    Set m = Runtime1.NewMessage
    m.ExtraHeader("Reply-To") = "Me@mascro.com"
    If (0 = m.Send("myaccount@usermail.com", "bul@bulstone.com", "Test", "Hello 
world!")) Then
        MsgBox "The message was successfully delivered "
    End If
End Sub
```

property Message.From as String

Retrieves or sets a value that indicates the email address of the sender.

Type	Description
String	A string expression that indicates the email address of the sender. The string can contain the name of the sender between "" and the email address between brackets <>, like this "Mike" <mike@margs.com> or simple like: mike@margs.com

Use From property to specify the email address of the sender. Also, you can use the From optional argument of the Send method to set up the sender's email address. Some SMTP servers doesn't accept any email address for the sender. Some of the SMTP servers checks if the domain of the email address exists. The following sample shows how to set the name of sender:

```
Dim WithEvents e As EMail

Private Sub e_EndSend(ByVal Msg As EXEMAILLibCtl.IMessage)
   Debug.Print e.Error(Msg.LastError)
End Sub

Private Sub Form_Load()
   Set e = Runtime1.NewEMail()
   Dim m As Message
   Set m = Runtime1.NewMessage
   Set m.Notifier = e
   m.From = """Mike Philips"" <mike@sodkex.com>"
   m.Send , "jhon@margo.com", "Hi", "Hi buddy!"
End Sub
```

property Message.Helo as String

Specifies the string being sent by HELO command to the e-mail server.

Type	Description
String	A String expression that indicates the string being set by HELO command.

By default, the Helo property indicates the name of the host, who sends the message. As a result, the receiver-SMTP will not have to perform MX resolution on this name in order to validate the HELO parameter. The HELO receiver MAY verify that the HELO parameter really corresponds to the IP address of the sender. However, the receiver MUST NOT refuse to accept a message, even if the sender's HELO command fails verification.

property Message.Host as String

Retrieves or sets a value that indicates the host's address where the message should be delivered.

Type	Description
String	A String expression that indicates the host where the message should be delivered.

By default, the message's Host is empty. If the Host is empty, the control queries the DNS for the SMTP servers where the email should be delivered. You can use the Host property to force control using a certain SMTP server. If the Host is not empty, the Send method does not query the DNS server for the mail exchange information. If the Host property is not empty, the component doesn't check if the host is valid. Use the LastError property to check the last error occurred.

The following sample shows how to use Host property:

```
Dim WithEvents e As EMail

Private Sub e_EndSend(ByVal Msg As EXEMAILLibCtl.IMessage)
   Debug.Print e.Error(Msg.LastError)
End Sub

Private Sub Form_Load()
   Set e = Runtime1.NewEMail()
   Dim m As Message
   Set m = Runtime1.NewMessage
   Set m.Notifier = e
   m.Host = "unknown.com"
   m.Send "me", "mike2@Unknown.com", "Test", "Hello world!"
End Sub
```

property Message.LastError as Long

Retrieves the last error occurred.

Type	Description
Long	A long expression that indicates the last error occurred.

Use the LastError to check whether the email message was successfully delivered. If the message failed, the LastError retrieves the last error code. If the message was successfully sent the LastError gets 0. Use the Error property to get the description for an error code. If the Notifer property points to an EMail object then the Send method retrieves 0. In this case, 0 means that the control has started delivering the message. Use EndSend event to check out if the message was delivered ok. If the Notifier property is set to nothing the Send method retrieves the last error code.

The following samples show how to check whether a message was successfully sent:

```
Dim WithEvents e As EMail

Private Sub e_EndSend(ByVal Msg As EXEMAILLibCtl.IMessage)
    If (Msg.LastError = 0) Then
        MsgBox "The message was delivered OK."
    End If
    Debug.Print e.Error(Msg.LastError)
End Sub

Private Sub Form_Load()
    Set e = Runtime1.NewEMail()
    Dim m As Message
    Set m = Runtime1.NewMessage
    Set m.Notifier = e
    m.Send "me", "mike2@Unknown.com", "Test", "Hello world!"
End Sub

Private Sub Form_Load()
    Dim m As Message
    Set m = Runtime1.NewMessage
    If (0 = m.Send("me", "mike2@Unknown.com", "Test", "Hello world!")) Then
        MsgBox "The message was delivered OK."
    End If
End Sub
```

property Message.LogonPassword as String

Retrieves or sets the value that indicates the login's password.

Type	Description
String	A string expression that specifies the login's password.

Some SMTP Servers require an username and a password. The value for the LogonUser and LogonPassword are sent to the SMTP server only if the AuthMethod property is AuthLogin or AuthPlain.

property Message.LogonUser as String

Retrieves or sets a value that indicates the login user name.

Type	Description
String	A string expression that indicates the login user name.

Some SMTP Servers require an username and a password. The value for the LogonUser and LogonPassword are sent to the SMTP server only if the AuthMethod property is AuthLogin or AuthPlain.

property Message.Notifier as EMail

Changes or gets the message's object notifier.

Type	Description
EMail	An EMail object that receives all notifications.

By default, the Notifier property is set to nothing. The Notifier property specifies the way how the message is going to be delivered: non-blocking or blocking mode. In non-blocking mode the control fires events through EMail object, and it doesn't block the application while delivering the message. In blocking mode, the control doesn't fire any event, and the application is blocked while the control delivers the message. If the Notifier property points to an EMail object, then all message's notifications are handled by EMail object' events. The following sample shows how to deliver an email message using non-blocking mode:

```
Dim WithEvents e As EMail

Private Sub e_EndSend(ByVal Msg As EXEMAILLibCtl.IMessage)
    If (Msg.LastError = 0) Then
        MsgBox "The message was delivered OK."
    End If
    Debug.Print e.Error(Msg.LastError)
End Sub

Private Sub Form_Load()
    Set e = Runtime1.NewEMail
    Dim m As Message
    Set m = Runtime1.NewMessage
    Set m.Notifier = e
    m.Send "me", "mike2@Unknown2.com", "Test", "Hello world!"
End Sub
```

The following sample shows how to deliver an email message using the blocking mode:

```
Private Sub Form_Load()
    Dim m As Message
    Set m = Runtime1.NewMessage
    If (0 = m.Send("me", "mike2@Unknown2.com", "Test", "Hello world!")) Then
        MsgBox "The message was delivered OK."
    End If
End Sub
```

property Message.Port as Long

Retrieves or sets the server's port.

Type	Description
Long	A long expression that indicates the SMTP server port.

By default, the Port property is 25. Some SMTP servers require another port to send data to.

property Message.Priority as PriorityEnum

Retrieves or sets a value that indicates the message's priority.

Type	Description
PriorityEnum	A PriorityEnum expression that indicates the message's priority.

Use the Priority property to change the message's priority. By default, the message's priority is Normal. The following sample shows how to send an high priority message:

```
Private Sub Form_Load()
    Dim m As Message
    Set m = Runtime1.NewMessage
    m.Priority = High
    If (0 = m.Send("me", "mike2@Unknown2.com", "Test", "Hello world!")) Then
        MsgBox "The message was delivered OK."
    End If
End Sub
```

method Message.Send ([From as Variant], [To as Variant], [Subject as Variant], [Message as Variant], [Attachement as Variant])

Sends the email message to the recipients.

Type	Description
From as Variant	An optional string expression that indicates the email address for the sender. If the parameter is missing, the control considers the From property been the sender's email address.
To as Variant	An optional string expression that indicates the recipients where the email message should be delivered. If the parameter is missing, the control considers the To property been the recipient email addresses. The list of recipients should be separated by commas.
Subject as Variant	An optional string expression that indicates the email message's subject. If the parameter is missing, the control considers the Subject property been the email message's subject.
Message as Variant	An optional string expression that indicates the email message's body as plain text. If the parameter is missing, the control considers the BodyText property being the email message's body text.
Attachement as Variant	An optional string expression that indicates the attachments of the email message. A String expression that indicates the files to attach as: • a list of files (including the full path) to attach, separated by semicolon • a multi-line string defining the file name to attach, with subsequent lines specifying the content of the file to attach (ability to attach a file with its content without physically having it) If the parameter is missing, the control considers the Attachment property being the attachments for the message. To include attachments in your email message, there are two methods you can use: • Attachment property: Set the Attachment property of your email message object. This property allows you to specify the files you want to attach directly within

your email composition process
- Attachment argument of Send method: Alternatively, when calling the Send method to send your email, you can pass an Attachment argument. This argument lets you include files as attachments at the time of sending the email

Return	Description
Long	A long expression that indicates the error's code. If the Notifier property points to an EMail object then value returned is 0

The Send method sends the email message to the To, Cc and Bcc recipients. To send an email message using blocking or non-blocking mode use Notifier property, before calling Send method. If the message's Notifer is set to nothing, the Send method retrieves the last error occurred. Use the Error property to get the description given the error code. If the Notifier property points to an EMail object the Send method retrieves 0, that means, that the control has started delivering the email message using non-blocking mode. Use the SendOnce property On True, to prevent overloading the mail server, when the message should be delivered to multiple recipients, so the message is sent only once, and the server is responsible for delivering the message to recipients.

The following sample shows how to send an email message using non-blocking way:

```
Dim WithEvents e As EMail

Private Sub e_EndSend(ByVal Msg As EXEMAILLibCtl.IMessage)
    If (Msg.LastError = 0) Then
        MsgBox "The message was delivered OK."
    End If
    Debug.Print e.Error(Msg.LastError)
End Sub

Private Sub Form_Load()
    Set e = Runtime1.NewEMail
    Dim m As Message
    Set m = Runtime1.NewMessage
    Set m.Notifier = e
    m.Send "me", "mike2@Unknown2.com", "Test", "Hello world!"
End Sub
```

The following sample shows how to send the email message using blocking mode:

```
Private Sub Form_Load()
    Dim m As Message
    Set m = Runtime1.NewMessage
    If (0 = m.Send("me", "mike2@Unknown2.com", "Test", "Hello world!")) Then
        MsgBox "The message was delivered OK."
    End If
End Sub
```

property Message.SendOnce as Boolean

Indicates whether the message is sent once to the mail server, for all recipients, and the server takes the responsibility to delivery the message per all assigned recipients.

Type	Description
Boolean	A Boolean expression that specifies whether the message is sent once to the mail server, for all recipients, and the server takes the responsibility to delivery the message per all assigned recipients.

By default, the SendOnce property is False, which indicates that the message is sent individually to each recipients. The Send method sends the email message to the To, Cc and Bcc recipients. Use the SendOnce property On True, to prevent overloading the mail server so the message is sent only once, and the server is responsible for delivering the message to recipients.

property Message.Subject as String

Retrieves or sets a value that indicates the subject of the email message.

Type	Description
String	A String expression that indicates the email message's subject.

Use the Subject property to set up your email message's subject. Changing the Subject, optional parameter of the Send method is equivalent with changing the Subject property.

The following sample shows how to send an email message with the Subject "hello":

```
Private Sub Form_Load()
    Dim m As Message
    Set m = Runtime1.NewMessage
    If (0 = m.Send("me", "mike2@Unknown2.com", "Hello")) Then
        MsgBox "The message was delivered OK."
    End If
End Sub
```

property Message.Timeout as Long

Specifies the amount of time (in seconds) the control will wait for the server response.

Type	Description
Long	A long expression that Specifies the amount of time (in seconds) the control will wait for the server response.

Use the Timeout property to increase or decrease the amount of time (in seconds) the control will wait for the server response. By default, the Timeout property is set to 5 minutes. -1 is used to be an infinite timeout. Be carefully when you use Timeout = -1, and blocking mode, because it may block indefinitely your application.

property Message.To as String

Retrieves or sets a value that indicates the recipient's email address, or a list of recipient's email addresses separated by commas.

Type	Description
String	A String expression that indicates the recipient's email addresses separated by commas.

Use the To property to specify the email address where you want to send the email message. You can set the To property, by using To optional parameter of the Send method.

The recipient's email addresses should be separated by commas, like in the following sample:

```
Private Sub Form_Load()
    Dim m As Message
    Set m = Runtime1.NewMessage
    If (0 = m.Send("me",
"mike1@Unknown2.com,mike2@Unknown2.com,mike3@Unknown2.com", "Hello")) Then
        MsgBox "The message was delivered OK."
    End If
End Sub
```

Runtime object

Tip The /COM object can be placed on a HTML page (with usage of the HTML object tag: <object classid="clsid:...">) using the class identifier: {963635DE-2172-450C-A133-6983D6185794}. The object's program identifier is: "ExEMail.Runtime". The /COM object module is: "ExEMail.dll"

The Runtime object provides licensed EMail or Message objects at runtime. Here's the list of supported properties:

Name	Description
NewEMail	Creates and initializes a licensed EMail object at runtime.
NewMessage	Creates and initializes a licensed Message object at runtime.
Version	Retrieves the control's version.

property Runtime.NewEMail as EMail

Creates and initializes a licensed EMail object at runtime.

Type	Description
EMail	An EMail object being created

Use the NewEMail property to create licensed EMail objects at runtime. If you deploy an application that uses CreateObject or New statements for creating new EMail objects, the application wont work on the machine where the ExEMail wasn't licensed. Instead, you have to use NewEmail and NewMessage properties to create licensed objects at runtime.

The following sample shows how to send an email using blocking mode:

```
Private Sub Form_Load()
    Dim m As Message
    Set m = Runtime1.NewMessage
    If (0 = m.Send("me",
"mike1@Unknown2.com,mike2@Unknown2.com,mike3@Unknown2.com", "Hello")) Then
        MsgBox "The message was delivered OK."
    End If
End Sub
```

The following sample shows how to send an email using non-blocking way:

```
Dim WithEvents e As EMail

Private Sub e_EndSend(ByVal Msg As EXEMAILLibCtl.IMessage)
    If (Msg.LastError = 0) Then
        MsgBox "The message was delivered OK."
    End If
    Debug.Print e.Error(Msg.LastError)
End Sub

Private Sub Form_Load()
    Set e = Runtime1.NewEMail
    Dim m As Message
    Set m = Runtime1.NewMessage
    Set m.Notifier = e
    m.Send "me",
"mike1@Unknown2.com,mike2@Unknown2.com,mike3@Unknown2.com", "Hello"
End Sub
```

property Runtime.NewMessage as Message

Creates and initializes a licensed Message object at runtime.

Type	Description
Message	A Message object being created.

Use the NewMessage to create licensed Message objects at runtime If you deploy an application that uses CreateObject or New statements for creating new Message objects, the application wont work on the machine where the ExEMail wasn't licensed. Instead, you have to use NewEmail and NewMessage properties to provide licensed objects at runtime. The following sample shows how to send an email using blocking mode:

```
Private Sub Form_Load()
    Dim m As Message
    Set m = Runtime1.NewMessage
    If (0 = m.Send("me",
"mike1@Unknown2.com,mike2@Unknown2.com,mike3@Unknown2.com", "Hello")) Then
        MsgBox "The message was delivered OK."
    End If
End Sub
```

The following sample shows how to send an email using non-blocking way:

```
Dim WithEvents e As EMail

Private Sub e_EndSend(ByVal Msg As EXEMAILLibCtl.IMessage)
    If (Msg.LastError = 0) Then
        MsgBox "The message was delivered OK."
    End If
    Debug.Print e.Error(Msg.LastError)
End Sub

Private Sub Form_Load()
    Set e = Runtime1.NewEMail
    Dim m As Message
    Set m = Runtime1.NewMessage
    Set m.Notifier = e
    m.Send "me",
"mike1@Unknown2.com,mike2@Unknown2.com,mike3@Unknown2.com", "Hello"
End Sub
```

property Runtime.Version as String

Retrieves the control's version.

Type	Description
String	A string expression that indicates the control's version.

The Version property specifies the version of the control that's running.

ExEMail events

Tip The /COM object can be placed on a HTML page (with usage of the HTML object tag: <object classid="clsid:...">) using the class identifier: {963635DE-2172-450C-A133-6983D6185794}. The object's program identifier is: "ExEMail.Runtime". The /COM object module is: "ExEMail.dll"

All of the ExEMail events are fired only if the Notifier property points to an EMail object. The following sample shows how to set the Notifier property:

```
Dim WithEvents e As EMail

Private Sub Form_Load()
    ' Creates an EMail object, that will receive Message notifications
    Set e = Runtime1.NewEMail()
    ' Creates a Message object
    Dim m As Message
    Set m = Runtime1.NewMessage()
    ' Message' notifications are sent through EMail object
    Set m.Notifier = e
    ' Sends a message to 2 users
    m.Send "me@malibu.com", "bula@bulat.com,robingo@calcuta.fom", "Test", "The
message has one attchament", "c:\winnt\system32\setup.exe"
End Sub
```

In order to use ExEMail events in C++, you can use the following class's header:

```
#if
!defined(AFX_EMAILNOTIFIER_H__73B49333_43C5_4E00_B3B8_44D124A2FB6C

#define
AFX_EMAILNOTIFIER_H__73B49333_43C5_4E00_B3B8_44D124A2FB6C__INCLUDE


#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include <comdef.h>
#include <atlbase.h>

struct __declspec(uuid("84f98dce-e10f-4002-a605-5c39baffc801"))
/* interface */ _IEMailEvents;
```

```
struct __declspec(uuid("84f98dce-e10f-4002-a605-5c39baffc801"))
_IEMailEvents : IDispatch
{
// Methods:
HRESULT BeginSend ( struct IMessage * Msg );
HRESULT StateChange ( struct IMessage * Msg, enum SendStateEnum
NewState );
HRESULT EndSend ( struct IMessage * Msg );
HRESULT Debug ( struct IMessage * Msg, _bstr_t Description );
HRESULT Data ( struct IMessage * Msg, double Percent,
VARIANT_BOOL * Cancel );
};

class EMailNotifier :
    public _IEMailEvents
{
public:

    // Constrcutor & Destrcutor
    EMailNotifier() : m_dwCookie( NULL ){
    };
    virtual ~EMailNotifier(){
    };

    // IUnknown
    STDMETHOD_(ULONG, AddRef)() {
        return 1;
    }

    STDMETHOD_(ULONG, Release)() {
        return 0;
    }

    STDMETHOD(QueryInterface)(REFIID iid, void** ppvObject) {
        if (!ppvObject)
            return E_POINTER;

        // check for the interfaces this object knows about
        if (iid == IID_IUnknown)
        {
            *ppvObject = (IUnknown*)this;
            AddRef();
            return S_OK;
        }
        if (iid == IID_IDispatch)
        {
            *ppvObject = (IDispatch*)this;
            AddRef();
            return S_OK;
        }
        if (iid == __uuidof(_IEMailEvents) )
        {
            *ppvObject = (_IEMailEvents*)this;
            AddRef();
            return S_OK;
        }

        // otherwise, incorrect IID, and thus error
        return E_NOINTERFACE;
    }

    // IDispatch
    STDMETHOD(GetTypeInfoCount)(unsigned int*) {
        return E_NOTIMPL;
    }
    STDMETHOD(GetTypeInfo)(unsigned int, LCID, ITypeInfo**) {
        return E_NOTIMPL;
    }

    STDMETHOD(GetIDsOfNames)(REFIID, LPOLESTR*, unsigned int,
LCID, DISPID*) {
        return E_NOTIMPL;
    }

    STDMETHOD(Invoke)(DISPID id, REFIID, LCID, unsigned short,
DISPPARAMS* p,
                    VARIANT*, EXCEPINFO*, unsigned int*) {
        switch ( id )
        {
            case 1: // BeginSend
            {
                BeginSend( (IMessage*)V_DISPATCH( &p->rgvarg[0] )
);
                break;
            }
            case 2: // StateChange
            {
                Sending( (IMessage*)V_DISPATCH( &p->rgvarg[1] ),
(SendStateEnum)V_I4( &p->rgvarg[0] ) );
                break;
            }
            case 3: // EndSend
            {
                EndSend( (IMessage*)V_DISPATCH( &p->rgvarg[0] )
);
                break;
            }
            case 4: // Debug
            {
                Debug( (IMessage*)V_DISPATCH( &p->rgvarg[1] ),
V_BSTR(&p->rgvarg[0] ) );
                break;
            }
            case 5: // Data
            {
                Data( (IMessage*)V_DISPATCH( &p->rgvarg[2] ),
V_R8(&p->rgvarg[1] ), V_BOOLREF(&p->rgvarg[0] ) );
                break;
            }
        }
        return S_OK;
    }

    // _IEMailEvents
virtual HRESULT BeginSend ( struct IMessage * Msg ) = NULL;
virtual HRESULT Sending ( struct IMessage * Msg, enum
SendStateEnum SendState ) = NULL;
virtual HRESULT EndSend ( struct IMessage * Msg ) = NULL;
virtual HRESULT Debug ( struct IMessage * Msg, _bstr_t
Description ) = NULL;
virtual HRESULT Data ( struct IMessage * Msg, double Percent,
VARIANT_BOOL * Cancel ) = NULL;

    // Function name    : Start
    // Description      : Starts sinking the pIEMail
    // Return type         : virtual void
    // Argument : IDispatch* pIEMail
    virtual HRESULT Start( IDispatch* pIEMail ) {
        if ( CComQIPtr<IConnectionPointContainer> spCPC( pIEMail
) )
        {
            CComPtr<IConnectionPoint> spCP;
            if ( SUCCEEDED( spCPC->FindConnectionPoint(
__uuidof(_IEMailEvents), &spCP ) ) )
                return spCP->Advise( this, &m_dwCookie );
        }
        return E_FAIL;
    }

    // Function name    : Stop
    // Description      : Stops sinking the pIEmail
    // Return type        : virtual void
    // Argument : IDispatch* pIEMail
    virtual HRESULT Stop( IDispatch* pIEMail ) {
        if ( CComQIPtr<IConnectionPointContainer> spCPC( pIEMail
) )
        {
            CComPtr<IConnectionPoint> spCP;
            if ( SUCCEEDED( spCPC->FindConnectionPoint(
__uuidof(_IEMailEvents), &spCP ) ) )
                return spCP->Unadvise( m_dwCookie );
        }
        return E_FAIL;
    }

protected:

    // Attributes
    DWORD m_dwCookie;
};

#endif //
!defined(AFX_EMAILNOTIFIER_H__73B49333_43C5_4E00_B3B8_44D124A2FB6C
```

Here's the list of supported events:

Name	Description
BeginSend	Fired just before starting sending the message.
Data	Occurs during sending the message's data.
Debug	Fired while the control communicates with the host.
EndSend	Fired after the control has done sending the message.
StateChange	Occurs during sending the message, when the connection's state is changing/changed.

event BeginSend (Msg as Message)

Fired just before starting sending the message.

Type	Description
Msg as Message	A Message object being sent.

The BeginSend event is fired when Send method starts sending the message. The BeginEnd event is fired only if the Notifier property points to the EMail object. Once that BeginSend message was fired, the Message object is not released until EndSend event is fired. Use the BeginSend event to notify your application when a Message starts to be send. The following sample shows how to send two email messages:

```
Dim WithEvents e As EMail

Private Sub e_BeginSend(ByVal Msg As EXEMAILLibCtl.IMessage)
    Debug.Print Msg.Subject & " message begins. "
End Sub

Private Sub e_EndSend(ByVal Msg As EXEMAILLibCtl.IMessage)
    Debug.Print Msg.Subject & " message ends."
End Sub

Private Sub e_Debug(ByVal Msg As EXEMAILLibCtl.IMessage, ByVal Description As String)
    Debug.Print Msg.Subject & " DEBUG: " & Description
End Sub
```

```
Private Sub Form_Load()
    ' Creates an EMail object, that will receive Message notifications
    Set e = Runtime1.NewEMail()
    ' Creates two Message objects
    Dim m1 As Message
    Set m1 = Runtime1.NewMessage()
    ' Message' notifications are sent through EMail object
    Set m1.Notifier = e
    Set m2 = Runtime1.NewMessage()
    ' Message' notifications are sent through EMail object
    Set m2.Notifier = e
    ' The email addresses are absolutely arbitrary!

    ' Sends a message to one user
    m1.Send "me@malibu.com", "user1@yahoo.com", "Test1", "TEST"

    ' Sends a message to 2 users
    m2.Send "me@malibu2.com", "user2@yahoooo.com,user3@cobra.com", "Test2",
"TEST"
End Sub
```

event Data (Msg as Message, Percent as Double, ByRef Cancel as Boolean)

Occurs during sending the message's data.

Type	Description
Msg as Message	A Message object being sent.
Percent as Double	A double value that indicates the percent of the data sent. The valid range is [0..100].
Cancel as Boolean	(By Reference) A boolean reference to let user cancels sending the message.

The Data event is fired each time when a new line of message's data was sent to the host. The Data event is fired only if the Notifier property points to the EMail object. The Percent value specifies the amount of data in % that has been already delivered. Use Data event to provides progress capabilities to your application. You can cancel sending the message's data by changing the Cancel value to True. We cannot guarante that the message can be canceled. It depends on SMPT server implementation. The following sample shows how to add progress capabilities to your form. In order to run the following sample your form requires a Microsoft ProgressBar ActiveX control, or something similar:

```
Dim WithEvents e As EMail

Private Sub e_BeginSend(ByVal Msg As EXEMAILLibCtl.IMessage)
   Debug.Print Msg.Subject & " message begins. "
End Sub

Private Sub e_Data(ByVal Msg As EXEMAILLibCtl.IMessage, ByVal Percent As Double,
Cancel As Boolean)
   ProgressBar1.Value = Percent
End Sub

Private Sub e_Debug(ByVal Msg As EXEMAILLibCtl.IMessage, ByVal Description As String)
   Debug.Print Msg.Subject & " DEBUG: " & Description
End Sub

Private Sub e_EndSend(ByVal Msg As EXEMAILLibCtl.IMessage)
   Debug.Print Msg.Subject & " message ends."
End Sub
```

```
Private Sub Form_Load()
    ProgressBar1.Min = 0
    ProgressBar1.Max = 100

    ' Creates an EMail object, that will receive Message notifications
    Set e = Runtime1.NewEMail()
    ' Creates a Message object
    Dim m As Message
    Set m = Runtime1.NewMessage()
    ' Message' notifications are sent through EMail object
    Set m.Notifier = e

    ' Sends a message to one user
    m.Send "me@malibu.com", "testttt@hotmail.com", "Test", "The message has one
attchament", "c:\winnt\system32\setup.exe"

End Sub
```

event Debug (Msg as Message, Description as String)

Fired while the control communicates with the host.

Type	Description
Msg as Message	A Message object being sent.
Description as String	A string expression that indicates the command sent to the host, or the reply message from the host. The commands, and replies are conformed to RFC 821, 822.

The Debug event is fired each time when the control sends a command to the SMTP server,or when the SMTP server replies. Also, the Debug event is fired when the control is trying to find the SMTP hosts responsible for the recipent's email address. For instance, if you are trying to send a message "testsss@hotmail.com", the control queries a DNS server for the SMTP hosts that are responsible for the domain email address. Once that DNS replies the SMTP servers, the control is trying one by one SMTP server until the message was delivered successfully. The Debug event shows how delivering the message goes. The following sample displays the SMTP server where the current message is delivered:

```
Dim WithEvents e As EMail

Private Sub e_Debug(ByVal Msg As EXEMAILLibCtl.IMessage, ByVal Description As
String)
    If (Description Like "Trying host:*") Then
        Debug.Print Msg.Subject & " - " & Description
    End If
End Sub

Private Sub Form_Load()

   ' Creates an EMail object, that will receive Message notifications
    Set e = Runtime1.NewEMail()
    ' Creates a Message object
    Dim m As Message
    Set m = Runtime1.NewMessage()
    ' Message' notifications are sent through EMail object
    Set m.Notifier = e

    ' Sends a message to one user
    m.Send "me@malibu.com", "testsss@hotmail.com", "Test", "The message has one
attchament", "c:\winnt\system32\setup.exe"
End Sub
```

event EndSend (Msg as Message)

Fired after the control has done sending the message.

Type	Description
Msg as Message	A Message object that contains the email message being sent to the SMTP server.

The EndSend event is fired when sending the email message is done. The EndSend event is fired only if the Notifier property points to the EMail object. After the EndSend event was fired, the EMail object releases the Message object. The Message's count reference is increased by BeginSend event, and it is decreased by the EndSend event. In order to check whether the message was delivered OK you must handle the EndSend event. To check the latest error occurs use the LastError property. Use Error property to get the description for an error code. The following sample shows how to check whether the message was successfully delivered:

```
Dim WithEvents e As EMail

Private Sub e_Debug(ByVal Msg As EXEMAILLibCtl.IMessage, ByVal Description As
String)
    Debug.Print Description
End Sub

Private Sub e_EndSend(ByVal Msg As EXEMAILLibCtl.IMessage)
    If (Msg.LastError = 0) Then
        MsgBox "The message '" & Msg.Subject & "' was was successfully delivered."
    Else
        MsgBox "The message '" & Msg.Subject & "' failed. " & e.Error(Msg.LastError)
    End If
End Sub

Private Sub Form_Load()

    ' Creates an EMail object, that will receive Message notifications
    Set e = Runtime1.NewEMail()
    ' Creates a Message object
    Dim m As Message
    Set m = Runtime1.NewMessage()
    ' Message' notifications are sent through EMail object
    Set m.Notifier = e

    ' Sends a message to one user
    m.Send "me@malibu.com", "robingo@calcuta.fom", "Test", "The message has one
attchament", "c:\winnt\system32\setup.exe"
End Sub
```

event StateChange (Msg as Message, NewState as SendStateEnum)

Occurs during sending the message, while the sending's state is changing/changed.

Type	Description
Msg as Message	A Message object that contains the email message being sent to the SMTP server.
NewState as SendStateEnum	A SendStateEnum expression that indicates the state of the sending operation.

The StateChange event is fired each time when the sending state is changing/changed. Use the Debug event to debug sending a message. Here are the steps that the ExEMail component follows in order to deliver an email message:

1. The control fires BeginSend() event
2. The control takes the domain recipent's email address. The control fires StateChange(...,LookupMX)
3. The control sends a query to the DNS server in order to get the SMTP hosts that are responsible for the domain.
4. Based on the DNS's response, the control is trying each SMTP server found to deliver the email message. The control fires Debug event, in order to let the application knows what's the host where the control is trying to deliver the email message. The control fires StateChange(...,Connecting)
5. Once that the control is connected to the SMTP server, the StateChange(...,Connected) is fired.
6. When the control says hello to the server. it fires StateChange(...,Opening)
7. The control fires StateChange(...,Opened), after server replies to the hello command.
8. The control prepares and sends the information about the sender and recipents email addresses.
9. If the server accepts the sender and the recipents, the control prepares the message's DATA to send it to the SMTP server. Before starting any data to the server, the control fires StateChange(...,Data). During sending message's data the control fires Data event.
10. Once that message's data was delivered, the control is going to close the connection. It fires StateChange(...,Disconnecting)
11. The client is going to inform the server that wants to close the connection. The control fires StateChange(...,Closing)
12. The client sends QUIT command to the server, and fires the StateChange(...,Closed)
13. The client is disconnected, and the control fires StateChange(...,Disconnected)
14. The control fires EndSend() event

The events are fired only if the Notifier property points to an EMail object. For instance the following sample displays the message's state during sending:

```
Dim WithEvents e As EMail

Private Sub e_StateChange(ByVal Msg As EXEMAILLibCtl.IMessage, ByVal NewState As
EXEMAILLibCtl.SendStateEnum)
    Select Case NewState
        Case SendStateEnum.LookupMX
            Debug.Print "LookupMX"
        Case SendStateEnum.Connected
            Debug.Print "Connected"
        Case SendStateEnum.Connecting
            Debug.Print "Connecting"
        Case SendStateEnum.Opening
            Debug.Print "Opening"
        Case SendStateEnum.Opened
            Debug.Print "Opened"
        Case SendStateEnum.Data
            Debug.Print "Data"
        Case SendStateEnum.Closing
            Debug.Print "Closing"
        Case SendStateEnum.Closed
            Debug.Print "Closed"
        Case SendStateEnum.Disconnecting
            Debug.Print "Disconnecting"
        Case SendStateEnum.Disconnected
            Debug.Print "Disconnected"
    End Select
End Sub

Private Sub Form_Load()

    ' Creates an EMail object, that will receive Message notifications
    Set e = Runtime1.NewEMail()
    ' Creates a Message object
    Dim m As Message
    Set m = Runtime1.NewMessage()
    ' Message' notifications are sent through EMail object
    Set m.Notifier = e
    ' Sends a message to one user
    m.Send "m@malibu.com", "marfa@cool.com", "Test", "The message has one
attchament", "c:\winnt\system32\setup.exe"
End Sub
```