Table of Contents

Part I Introduction 1

1 Overview ... 1
2 What's new in version 4.2 ... 1
3 What's new in version 4.1 ... 1
4 What's new in version 4.0 ... 1
5 What's new in version 3.0 ... 2
6 What's new in version 2.0 ... 3
7 Legal Information ... 4
8 System Requirements ... 5
9 Redistributable Code ... 5
10 Version History ... 7

Part II Samples and Tutorials 8

1 C# ... 8
- Using AddEmail in C# projects ... 8
- Samples ... 10

2 VB.NET ... 11
- Using AddEmail in VB.NET projects ... 11
- Samples ... 13

3 ASP.NET - C# ... 14
- Using AddEmail in ASP.NET / C# projects ... 14
- Samples ... 17

4 ASP.NET - VB ... 17
- Using AddEmail in ASP.NET / VB projects ... 17
- Samples ... 20

5 VBA (Microsoft Office) ... 20
- Using AddEmail in VBA projects ... 20
- Samples ... 22

6 C++/MFC/ATL ... 23
- Using AddEmail in C++ projects ... 23
- Samples ... 25

7 VB6 ... 26
- Using AddEmail in VB6 projects ... 26
- Samples ... 28

8 VBScript ... 29
- Using AddEmail in VBScript projects ... 29
- Samples ... 31

9 JScript ... 31
- Using AddEmail in JScript projects ... 31
- Samples ... 33

10 ASP - VBScript ... 34
- Using AddEmail in ASP / VBScript projects ... 34

Samples ... 36
11 ASP - JScript ... 37
Using AddEmail in ASP / JScript projects ... 37
Samples ... 40
12 Delphi ... 40
Using AddEmail in Delphi projects ... 40
Samples ... 41
13 Visual FoxPro ... 42
Using AddEmail in FoxPro projects ... 42
14 PowerBuilder ... 43
Using AddEmail in PowerBuilder projects ... 43
Part III Reference 46
1 SmtpMail ... 46
Overview ... 46
SmtpServer Property ... 47
SmtpPort Property ... 48
SmtpUsername Property ... 48
SmtpPassword Property ... 49
SmtpSSL Property ... 50
SmtpSPA Property ... 50
MaxThreads Property ... 51
SenderHostname Property ... 52
SimpleSend Method ... 52
SimpleSendAttachment Method ... 54
SimpleSendHtml Method ... 56
SerialNumber Property ... 58
ConnectAttempts Property ... 58
ReplyTimeout Property ... 59
Send Method ... 60
SendAsync Method ... 61
GetStatus Method ... 62
GetErrorCode Method ... 64
GetErrorDescription Method ... 65
Cancel Method ... 66
CancelAll Method ... 67
GetLastErrorCode Method ... 68
GetLastErrorDescription Method ... 69
GetQueueSize Method ... 70
OnStatusChange Event ... 71
OnProgress Event ... 72
OnStatusChangeHandler Property ... 73
OnProgressHandler Property ... 73
2 MailMessage ... 73
Overview ... 73
Sender Property ... 75
ReplyTo Property ... 76
MessageSubject Property ... 76
MessageBody Property ... 77
MessageBodyFormat Property ... 78
MessageBodyEncoding Property ... 78
MessageBodyCharset Property ... 79

AltMessageBody Property ... 80
AltMessageBodyEncoding Property ... 80
AltMessageBodyCharset Property ... 81
MessagePriority Property ... 82
MessageId Property ... 82
MessageBodyFile Property ... 83
RequestReadReceipt Property ... 84
AddRecipient Method ... 84
AddCcRecipient Method ... 85
AddBccRecipient Method ... 87
AddAttachment Method ... 88
AddHeader Method ... 89
ClearAttachments Method ... 90
ClearRecipients Method ... 91
ClearCcRecipients Method ... 92
ClearBccRecipients Method ... 93
ImportHTML Method ... 94

3 MailAttachment ... 95

Overview ... 95
File Property ... 96
Name Property ... 97
Encoding Property ... 97
ContentType Property ... 98
Charset Property ... 98
Inline Property ... 99
ContentId Property ... 100
LoadFromMemory Method ... 100

4 MailAddress ... 101

Overview ... 101
Address Property ... 102
Name Property ... 102

5 MailFormat ... 103

6 MailEncoding ... 103

7 MailPriority ... 103

8 MailStatus ... 104

Index 0

I Introduction

1.1 Overview

Overview of AddEmail ActiveX

The AddEmail ActiveX enables your application to send e-mail messages. AddEmail was designed to work with all development environments that support COM objects, ActiveX or OCX controls. Easy-to-use and flexible object model allows application developers to implement e-mail support without a time-consuming learning curve. AddEmail was developed for today's most demanding applications and was thoroughly tested to ensure enterprise-level reliability and stability. The AddEmail ActiveX license agreement allows you to embed and distribute AddEmail within your application, royalty free. Main features of AddEmail ActiveX are:

- Send e-mails using SMTP and ESMTP protocols with optional authentication and SSL encryption.
- Send text or HTML e-mails with any number of attachments.
- Create or import HTML e-mails with embedded images.
- Supports e-mails with multiple recipients in To, Cc and Bcc fields.
- Sends large e-mails asynchronously without blocking your application.
- Progress and status events allow to implement user-friendly UI: progress indicator, cancel button etc.
- Sends several e-mails simultaneously without any multi-threading support in your application (AddEmail Enterprise version only).
- Sends e-mails directly to recipients' mail servers without using SMTP server of your organization or internet provider (AddEmail Enterprise version only).

Developer Support

Please visit **Traysoft Developer Tools** website at www.traysoft.com for the latest version of this manual. You can also find additional samples, news, updates and other AddEmail-related information on our website.

If you have any questions or problems please submit a support request or send an e-mail to devtools@traysoft.com. We respond to all inquires within 1-2 business days.

1.2 What's new in version 4.2

AddEmail ActiveX 4.2 resolves an issue affecting TLS 1.2 negotiation on some systems. New version also fixes intermittent decryption errors affecting some customers who use Office 365 mail service.

1.3 What's new in version 4.1

AddEmail ActiveX 4.1 improves compatibility with the latest versions of Transport Layer Security protocol TLS 1.1 and 1.2. This version resolves connection issues with Office 365 and Outlook mail servers.

1.4 What's new in version 4.0

AddEmail ActiveX 4.0 release adds full 64-bit support for all versions of Windows. Version 4.0 is fully

compatible with previous version 3.0. All applications developed with version 3.0 will work with AddEmail ActiveX 4.0 without any modifications.

Windows 10, Windows 8, Windows Server 2016 and Windows Server 2012 compatibility

AddEmail ActiveX 4.0 is fully compatible with all editions of Windows 10, Windows 8, Windows Server 2016 and Windows Server 2012, 64-bit (x64) and 32-bit (x86) versions. Version 4.0 includes 2 version of AddEmail ActiveX DLL:

AddEmail.dll - 32-bit COM DLL for 32-bit applications or .NET "Any CPU" applications running on 32-bit OS
AddEmail64.dll - 64-bit COM DLL for 64-bit applications or .NET "Any CPU" applications running on 64-bit OS

Visual Studio 2017, 2015, 2013 and 2012 support

We added support for all latest versions of Visual Studio. AddEmail ActiveX is now compatible with all Visual Studio versions, from the latest Visual Studio 2017 to the 15-year old Visual Studio .NET 2002. All C#, VB.NET and ASP.NET samples have been updated to .NET 4 and ASP.NET 4. Samples for older versions of .NET framework can be installed separately if needed.

Office 2016, 2013 and 2010 support

Version 4.0 supports all 32-bit and 64-bit versions of Office 2016, Office 2013 and Office 2010. We added new samples showing how to send emails from Microsoft Access and Microsoft Excel. The same VBA code can be used in other Office applications as well.

1.5 What's new in version 3.0

AddEmail ActiveX 3.0 adds several new features and improvements while maintaining complete compatibility with previous versions. Source code written for AddEmail ActiveX 1.x and 2.x should compile and work with AddEmail ActiveX 3.0 without any modifications. New features added to version 3.0 are listed below. For a complete list of changes please refer to the Version History.

Windows 7 and Windows Server 2008 R2 compatibility

AddEmail ActiveX 3.0 is fully compatible with all editions of Windows 7 and Windows Server 2008 R2, 64-bit and 32-bit versions. On 64-bit OS please use 32-bit version of regsvr32 to register AddEmail.dll on the computer. Run command prompt as an Administrator and type in:

```
c:\windows\syswow64\regsvr32 c:\addemail\addemail.dll
```

(assuming you copied AddEmail.dll to C:\AddEmail\ folder). You should see a message saying that the dll was registered successfully.

Automatic import of HTML with embedded images

With new version your program can automatically create HTML emails with embedded images by importing HTML files from disk. AddEmail imports specified HTML file, adds all images referenced in the HTML as inline attachments and modifies HTML as needed. Please refer to ImportHTML and SimpleSendHtml topics for more information.

SimpleSendXXX methods allow to specify sender's and recipients' names

New version adds support for "First LastName <name@domain.com>" format to SimpleSend, SimpleSendHtml and SimpleSendAttachment methods. Now you can specify names in addition to e-mail addresses for sender and recipients in SimpleSendXXX methods.

Support for Unicode (non-ASCII) attachment names

New version supports Unicode characters in MailAttachment.File and MailAttachment.Name properties. AddEmail will automatically encode attachment name if it has any non-ASCII characters.

ReplyTimeout property

New property has been added to specify how long AddEmail waits for a response from SMTP server before reporting timeout error. Please refer to ReplyTimeout topic for more information.

1.6 What's new in version 2.0

AddEmail ActiveX 2.0 adds many new features and improvements while maintaining complete compatibility with previous versions. Source code written for AddEmail ActiveX 1.x should compile and work with AddEmail ActiveX 2.0 without any modifications. New features added to version 2.0 are listed below.

Secure connection (TLS/SSL) support

Version 2.0 supports encrypted (TLS/SSL) connection to SMTP servers. Set SmtpSSL property to *True* if you want to use TLS/SSL connection, for example when using *smtp.gmail.com* server on port 465.

Secure Password Authentication (SPA) support

Version 2.0 supports Secure Password Authentication (SPA). If your SMTP server requires SPA please set SmtpSPA property to *True* in order to use SPA authentication method. Please note that some servers are configured to require SmtpUsername in *username@domain.com* format for SPA authentication.

Windows Vista support

Version 2.0 was extensively tested on Windows Vista and supports all Windows Vista editions. Please note that new security feature of Windows Vista called User Account Protection (UAP) prevents regular programs from writing to the HKEY_CLASSES_ROOT registry branch. Because of that, administrator privileges are required to register AddEmail.dll on Windows Vista. Only registration requires administrator privileges. After AddEmail.dll is registered on the target computer, it can be used to send e-mails from any program without administrator privileges.

New features of SimpleSend methods

Version 2.0 adds new SimpleSendAttachment method which allows to send text or HTML e-mails with any number of attachments to multiple recipients using just one method call. SimpleSend method also was modified to support text and HTML e-mails with multiple recipients. Please refer to the SimpleSend and SimpleSendAttachment topics in this manual for more information.

Improved Unicode support

Version 2.0 automatically detects Unicode characters and creates Unicode e-mails even if MessageBodyCharset property is not set. In addition, version 2.0 supports Unicode in the subject or in the body of the message when you use SimpleSend or SimpleSendAttachment.

Improved SMTP engine for sending e-mails directly (Enterprise version only)

SMTP engine in version 2.0 was updated to improve speed and compatibility when sending e-mails directly without using SMTP server of your organization or internet provider. New SenderHostname property allows to specify fully-qualified domain name of the sending computer. Please refer to the SenderHostname topic in this manual for more information.

New methods for obtaining error code and error description

GetLastErrorCode and GetLastErrorDescription methods were added in version 2.0 to improve compatibility with programming languages that do not support out parameters, e.g. JavaScript. These methods allow to obtain error code and error description of the last synchronous send operation.

Improved MIME engine

New MIME engine in version 2.0 creates e-mails that closely resemble e-mails created by Outlook, reducing probability that e-mails sent using AddEmail will be blocked by a spam filter.

Read receipt support

With previous versions of AddEmail custom header had to be added to the message to request a read receipt. To request read receipt with version 2.0 set RequestReadReceipt property of MailMessage object to *True*.

1.7 Legal Information

Legal Information for AddEmail 4.2 User Manual

While every precaution has been taken in the preparation of this document, Traysoft Inc. assumes no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall Traysoft Inc. be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Information in this document is subject to change without notice and does not represent a commitment on the part of Traysoft Inc. Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, unless expressly permitted by Traysoft Inc.

Traysoft Inc. may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property rights.

1.8 System Requirements

Operating Systems

Supported operating systems are:

- Microsoft Windows 11 64-bit (x64) and 32-bit (x86) versions
- Microsoft Windows 10 64-bit (x64) and 32-bit (x86) versions
- Microsoft Windows 8 64-bit (x64) and 32-bit (x86) versions
- Microsoft Windows 7 64-bit (x64) and 32-bit (x86) versions
- Microsoft Windows Vista 64-bit (x64) and 32-bit (x86) versions
- Microsoft Windows XP all editions
- Microsoft Windows Server 2003 - 2022
- Microsoft Windows 2000 Professional Workstation or Server

Development Environment

AddEmail can be used with any development environment that supports COM (ActiveX) objects, including:

- Microsoft Visual Studio 2002 - 2022
- .NET Framework 1.0 - 4.8
- ASP and ASP.NET
- Microsoft Office 365, Office 2010 - 2016, Office 2000 - 2007 and Office XP
- Microsoft Visual C++ 5.0 and 6.0
- Microsoft Visual Basic 5.0 and 6.0
- Microsoft Visual FoxPro
- Embarcadero/Borland C++ Builder
- Embarcadero/Borland Delphi
- Appeon/Sybase PowerBuilder

1.9 Redistributable Code

The Redistributable Code is the property of Traysoft Inc. and its suppliers and is protected by copyright law and international treaty provisions. You are authorized to make and use copies of the Redistributable Code either as part of the application in which you received the Redistributable Code, or in conjunction with the application for which its use is intended. Except as expressly provided in the End User License Agreement, you are not authorized to reproduce and distribute the Redistributable Code. Traysoft reserves all rights not expressly granted. You may not reverse engineer, decompile, or disassemble the Redistributable Code, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.

THE REDISTRIBUTABLE CODE IS PROVIDED TO YOU "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE. YOU ASSUME THE ENTIRE RISK AS TO THE ACCURACY AND THE USE OF THE REDISTRIBUTABLE CODE. TRAYSOFT SHALL NOT BE LIABLE FOR ANY DAMAGES WHATSOEVER ARISING OUT OF THE USE OF OR INABILITY TO USE THE REDISTRIBUTABLE CODE, EVEN IF TRAYSOFT HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Redistributable Code

Redistributable Code is identified as the following files and all of the files can be found at the following location:

```
Traysoft\AddEmail\Redistr\
```

Files:

AddEmail.dll - COM DLL that contains all AddEmail objects. AddEmail.dll is a self-registered COM DLL. If you are using setup builder such as InstallShield or Wise Installation Studio, simply add AddEmail.dll to the setup project and mark AddEmail.dll as "Self-register".

AddEmail64.dll - 64-bit version of AddEmail COM DLL. Use AddEmail64.dll for 64-bit applications or .NET "Any CPU" applications on 64-bit OS.

RegAddEmail.bat - Batch file for manual registration of *AddEmail.dll* and *AddEmail64.dll*. To register on a target computer copy *AddEmail.dll*, *AddEmail64.dll* and *RegAddEmail.bat* to the target computer then execute RegAddEmail.bat. On Windows Vista and later versions of Windows, please right-click on *RegAddEmail.bat* and select "Run as Administrator" then click "Allow".

1.10 Version History

Version 4.2

- Fixed an issue affecting TLS 1.2 negotiation on some systems.
- Fixed decryption errors with Office 365 mail service.

Version 4.1

- Improved compatibility with mail servers that require TLS 1.1 or TLS 1.2 encrypted connection.

Version 4.0

- Native 64-bit x64 version of AddEmail for 64-bit applications.
- Full support for Windows 10, Windows 8, Windows Server 2016 and Windows Server 2012.
- Support for Microsoft Office 2016, 2013 and 2010.
- New ASP.NET, C#, VB.NET, Access and Excel samples.

Version 3.0

- Full Windows 7 and Windows Server 2008 R2 support.
- Added HTML import. Now HTML emails with embedded images can be created automatically by importing HTML file from disk.
- Added support for "First LastName <name@domain.com>" format to all SimpleSendXXX methods.
- Added support for Unicode (non-ASCII) attachment names.
- Added ReplyTimeout property to specify how long AddEmail waits for a response from SMTP server before reporting timeout error.
- Improved error reporting for socket errors and SMTP protocol errors.
- Added checks for invalid symbols in e-mail addresses.
- Updated documentation and added new samples.

Version 2.1.1

- Added checks for proper format of sender's hostname when it's generated from computer name.
- Fixed a bug that was causing InitializeSecurityContext error with some SSL/TLS servers.

Version 2.1

- Added ConnectAttempts property to specify number of times AddEmail tries to connect to the SMTP server before reporting connection error.
- Added GetQueueSize method which returns number of e-mails queued for sending.
- Fixed a problem that prevented AddEmail from establishing SSL/TLS connection to some SMTP servers such as smtp.bizmail.yahoo.com and smtp.att.yahoo.com.
- Now AddEmail doesn't return timeout error if SMTP server takes a very long time to process an e-mail with large attachments.
- Added SetSender and SetReplyTo methods in order to support PowerBuilder 6.

Version 2.0.2

- Added support for custom From, To and Cc headers.
- Fixed a problem with text attachments in SimpleSendAttachment.

Version 2.0.1

- Improved SSL/TLS support.
- Updated C# 2005, VB.NET 2005 and ASP.NET 2.0 samples.

II Samples and Tutorials

2.1 C#

2.1.1 Using AddEmail in C# projects

To use AddEmail in your C# project perform the following steps:

1. Add a reference to the AddEmail library: In Visual Studio main menu, select View -> Solution Explorer to open Solution Explorer. Right-click on the project that will use AddEmail and select Add Reference from the pop-up menu. Click COM tab, find and select **AddEmail 4.0 Type Library**, click Select button then OK button.

2. Declare a variable that will hold a reference to the SmtpMail object:

```
private AddEmailLib.SmtpMailClass objSmtpMail;
```

3. Create an instance of the SmtpMail object in the constructor of your class:

```
objSmtpMail = new AddEmailLib.SmtpMailClass();
```

4. Set SMTP server address, port, username and password:

```
objSmtpMail.SmtpServer = "mail.myserver.com";
objSmtpMail.SmtpPort = 25;
objSmtpMail.SmtpUsername = "jsmith";
objSmtpMail.SmtpPassword = "mypassword";
```

5a. Send an e-mail synchronously using SimpleSend or SimpleSendAttachment:

```
string strError;
int resultCode = objSmtpMail.SimpleSend("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", out strError);
if (resultCode == 0)
{
  // E-mail was sent successfully
}
else
{
  // Send failed, strError contains detailed error description
}
```

5b. Alternatively, create and fill MailMessage object and send it synchronously using Send or asynchronously using SendAsync:

```
// Create message and setup subject and body
MailMessageClass objMessage = new AddEmailLib.MailMessageClass();
objMessage.MessageSubject = "test";
objMessage.MessageBody = "Test message";

// Setup sender
MailAddressClass objSender = new MailAddressClass();
objSender.Address = "jsmith@myserver.com";
objSender.Name = "Jane Smith";
objMessage.Sender = objSender;

// Setup first recipient
MailAddressClass objRecipient = new MailAddressClass();
objRecipient.Address = "jane@someserver.com";
objRecipient.Name = "Jane Smith";
objMessage.AddRecipient(objRecipient);

// Setup second recipient
objRecipient = new MailAddressClass();
objRecipient.Address = "james@someserver.com";
objRecipient.Name = "James Smith";
objMessage.AddRecipient(objRecipient);

// Send prepared message synchronously
string strError;
int resultCode = objSmtpMail.Send(objMessage, true, out strError);
if (resultCode == 0)
{
  // E-mail was sent successfully
}
else
{
  // Send failed, strError contains detailed error description
}

// Alternatively, send prepared message asynchronously
//int messageNumber = objSmtpMail.SendAsync(objMessage, true);
```

6. If the e-mail is sent asynchronously your application will need to process events to find out when the email was sent successfully or send operation failed. To process events from the SmtpMail object declare event handlers and subscribe to events:

```
private void OnStatusChange(int messageNumber, int newStatus)
{
    // Event processing code
}

private void OnProgress(int messageNumber, int bytesSent, int bytesTotal)
{
    // Event processing code
}

// Subscribe to events
objSmtpMail.OnStatusChange += new
_ISmtpMailEvents_OnStatusChangeEventHandler(OnStatusChange);
objSmtpMail.OnProgress += new _ISmtpMailEvents_OnProgressEventHandler(OnProgress);
```

Please refer to the Reference section of this manual for detailed description of AddEmail objects, methods and properties. Included C# samples provide code snapshots for common operations such as sending text e-mails, sending HTML e-mails, adding attachments to e-mails, creating HTML e-mails with embedded images.

2.1.2 Samples

AddEmail includes the following C# samples:

SimpleSend Sample

Location: *AddEmail\Samples\C# 2002-2003\SimpleSend*
AddEmail\Samples\C# 2005-2008\SimpleSend
AddEmail\Samples\C# 2010-2017\SimpleSend

SimpleSend sample demonstrates how to send text or HTML e-mails synchronously using SimpleSend.

SimpleSendAttachment Sample

Location: *AddEmail\Samples\C# 2002-2003\SimpleSendAttachment*
AddEmail\Samples\C# 2005-2008\SimpleSendAttachment
AddEmail\Samples\C# 2010-2017\SimpleSendAttachment

SimpleSendAttachment sample demonstrates how to send text or HTML e-mails with attachments synchronously using SimpleSendAttachment.

HtmlMail Sample

Location: *AddEmail\Samples\C# 2002-2003\HtmlMail*
AddEmail\Samples\C# 2005-2008\HtmlMail
AddEmail\Samples\C# 2010-2017\HtmlMail

HtmlMail sample demonstrates how to send HTML e-mails with attachments using Send. This sample also shows how to set alternative text-only body of e-mail.

ImportHtmlWithImages Sample

Location: *AddEmail\Samples\C# 2002-2003\ImportHtmlWithImages*
AddEmail\Samples\C# 2005-2008\ImportHtmlWithImages
AddEmail\Samples\C# 2010-2017\ImportHtmlWithImages

ImportHtmlWithImages sample demonstrates how to import HTML file with embedded images using ImportHTML.

EmbeddedImages Sample

Location: *AddEmail\Samples\C# 2002-2003\EmbeddedImages*
AddEmail\Samples\C# 2005-2008\EmbeddedImages
AddEmail\Samples\C# 2010-2017\EmbeddedImages

EmbeddedImages sample demonstrates how to create and send HTML e-mails with embedded images using Send.

EnterpriseMail Sample

Location: *AddEmail\Samples\C# 2002-2003\EnterpriseMail*
AddEmail\Samples\C# 2005-2008\EnterpriseMail
AddEmail\Samples\C# 2010-2017\EnterpriseMail

EnterpriseMail sample demonstrates features of AddEmail Enterprise version. It shows how to queue and send multiple e-mails simultaneously using AddEmail multi-threading support. This sample also shows how to process events from SmtpMail object. In addition, this sample shows how to send e-mails directly to recipients' mail servers without using outgoing mail server of your organization or internet provider.

UnicodeMail Sample

Location: *AddEmail\Samples\C# 2002-2003\UnicodeMail*
AddEmail\Samples\C# 2005-2008\UnicodeMail
AddEmail\Samples\C# 2010-2017\UnicodeMail

UnicodeMail sample demonstrates how to create and send e-mails that contain Unicode characters. Unicode can be used in the body of a message, in the subject field, and in the names of sender and recipients.

SendAsync Sample

Location: *AddEmail\Samples\C# 2002-2003\SendAsync*
AddEmail\Samples\C# 2005-2008\SendAsync
AddEmail\Samples\C# 2010-2017\SendAsync

SendAsync sample demonstrates how to send HTML e-mails with attachments asynchronously using SendAsync. This sample also shows how to process events from SmtpMail object.

2.2 VB.NET

2.2.1 Using AddEmail in VB.NET projects

To use AddEmail in your Visual Basic .NET project perform the following steps:

1. Add a reference to the AddEmail library: In Visual Studio main menu, select Project -> Add Reference. Click COM tab, find and select **AddEmail 4.0 Type Library**, click Select button then OK button.

2. Declare a variable that will hold a reference to the SmtpMail object and create an instance:

```
Dim WithEvents objSmtpMail As New AddEmailLib.SmtpMailClass()
```

3. Set SMTP server address, port, username and password:

```
objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpPort = 25
objSmtpMail.SmtpUsername = "jsmith"
```

```
objSmtpMail.SmtpPassword = "mypassword"
```

4a. Send an e-mail synchronously using SimpleSend or SimpleSendAttachment:

```
Dim strError As String
Dim resultCode As Integer
resultCode = objSmtpMail.SimpleSend("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", strError)
If resultCode = 0 Then
  ' E-mail was sent successfully
Else
  ' Send failed, strError contains detailed error description
End If
```

4b. Alternatively, create and fill MailMessage object and send it synchronously using Send or asynchronously using SendAsync method:

```
' Create message and setup subject and body
Dim objMessage As New AddEmailLib.MailMessageClass()
objMessage.MessageSubject = "test"
objMessage.MessageBody = "Test message"

' Setup sender
Dim objSender As New AddEmailLib.MailAddressClass()
objSender.Address = "jsmith@myserver.com"
objSender.Name = "John Smith"
objMessage.Sender = objSender

' Setup first recipient
Dim objRecipient As AddEmailLib.MailAddressClass
objRecipient = New AddEmailLib.MailAddressClass()
objRecipient.Address = "jane@someserver.com"
objRecipient.Name = "Jane Smith"
objMessage.AddRecipient(objRecipient)

' Setup second recipient
objRecipient = New AddEmailLib.MailAddressClass()
objRecipient.Address = "james@someserver.com"
objRecipient.Name = "James Smith"
objMessage.AddRecipient(objRecipient)

' Send prepared message synchronously
Dim strError As String
Dim resultCode As Integer
resultCode = objSmtpMail.Send(objMessage, True, strError)
If resultCode = 0 Then
  ' E-mail was sent successfully
Else
  ' Send failed, strError contains detailed error description
End If

' Alternatively, send prepared message asynchronously
'Dim messageNumber As Integer
'messageNumber = objSmtpMail.SendAsync(objMessage, True)
```

5. If the e-mail is sent asynchronously your application will need to process events to find out when the

email was sent successfully or send operation failed. Declare event handlers to process events from the SmtpMail object :

```
Private Sub OnStatusChange(ByVal messageNumber As Integer, ByVal newStatus As Integer)
Handles objSmtpMail.OnStatusChange
    ' Event processing code
End Sub

Private Sub OnProgress(ByVal messageNumber As Integer, ByVal bytesSent As Integer,
ByVal bytesTotal As Integer) Handles objSmtpMail.OnProgress
    ' Event processing code
End Sub
```

Please refer to the Reference section of this manual for detailed description of AddEmail objects, methods and properties. Included Visual Basic .NET samples provide code snapshots for common operations such as sending text e-mails, sending HTML e-mails, adding attachments to e-mails, creating HTML e-mails with embedded images.

2.2.2 Samples

AddEmail includes the following Visual Basic .NET samples:

SimpleSend Sample

Location: *AddEmail\Samples\VB.NET 2002-2003\SimpleSend*
AddEmail\Samples\VB.NET 2005-2008\SimpleSend
AddEmail\Samples\VB.NET 2010-2017\SimpleSend

SimpleSend sample demonstrates how to send text or HTML e-mails synchronously using SimpleSend.

SimpleSendAttachment Sample

Location: *AddEmail\Samples\VB.NET 2002-2003\SimpleSendAttachment*
AddEmail\Samples\VB.NET 2005-2008\SimpleSendAttachment
AddEmail\Samples\VB.NET 2010-2017\SimpleSendAttachment

SimpleSendAttachment sample demonstrates how to send text or HTML e-mails with attachments synchronously using SimpleSendAttachment.

HtmlMail Sample

Location: *AddEmail\Samples\VB.NET 2002-2003\HtmlMail*
AddEmail\Samples\VB.NET 2005-2008\HtmlMail
AddEmail\Samples\VB.NET 2010-2017\HtmlMail

HtmlMail sample demonstrates how to send HTML e-mails with attachments using Send. This sample also shows how to set alternative text-only body of e-mail.

ImportHtmlWithImages Sample

Location: *AddEmail\Samples\VB.NET 2002-2003\ImportHtmlWithImages*

AddEmail\Samples\VB.NET 2005-2008\ImportHtmlWithImages
AddEmail\Samples\VB.NET 2010-2017\ImportHtmlWithImages

ImportHtmlWithImages sample demonstrates how to import HTML file with embedded images using ImportHTML.

EmbeddedImages Sample

Location: *AddEmail\Samples\VB.NET 2002-2003\EmbeddedImages*
AddEmail\Samples\VB.NET 2005-2008\EmbeddedImages
AddEmail\Samples\VB.NET 2010-2017\EmbeddedImages

EmbeddedImages sample demonstrates how to create and send HTML e-mails with embedded images using Send.

EnterpriseMail Sample

Location: *AddEmail\Samples\VB.NET 2002-2003\EnterpriseMail*
AddEmail\Samples\VB.NET 2005-2008\EnterpriseMail
AddEmail\Samples\VB.NET 2010-2017\EnterpriseMail

EnterpriseMail sample demonstrates features of AddEmail Enterprise version. It shows how to queue and send multiple e-mails simultaneously using AddEmail multi-threading support. This sample also shows how to process events from SmtpMail object. In addition, this sample shows how to send e-mails directly to recipients' mail servers without using outgoing mail server of your organization or internet provider.

UnicodeMail Sample

Location: *AddEmail\Samples\VB.NET 2002-2003\UnicodeMail*
AddEmail\Samples\VB.NET 2005-2008\UnicodeMail
AddEmail\Samples\VB.NET 2010-2017\UnicodeMail

UnicodeMail sample demonstrates how to create and send e-mails that contain Unicode characters. Unicode can be used in the body of a message, in the subject field, and in the names of sender and recipients.

SendAsync Sample

Location: *AddEmail\Samples\VB.NET 2002-2003\SendAsync*
AddEmail\Samples\VB.NET 2005-2008\SendAsync
AddEmail\Samples\VB.NET 2010-2017\SendAsync

SendAsync sample demonstrates how to send HTML e-mails with attachments asynchronously using SendAsync. This sample also shows how to process events from SmtpMail object.

2.3 ASP.NET - C#

2.3.1 Using AddEmail in ASP.NET / C# projects

To use AddEmail in your ASP.NET / C# project perform the following steps:

1. Add a reference to the AddEmail library: In Visual Studio main menu, select View -> Solution Explorer to open Solution Explorer. Right-click on the project that will use AddEmail and select Add Reference from the pop-up menu. Click COM tab, find and select **AddEmail 4.0 Type Library**, click Select button then OK button.

2. Declare a variable that will hold a reference to the SmtpMail object:

```
AddEmailLib.SmtpMailClass objSmtpMail;
```

3a. Create an instance of the SmtpMail object:

```
objSmtpMail = new AddEmailLib.SmtpMailClass();
```

3b. If you are sending e-mails asynchronously you will need to access the same SmtpMail object on more than one page. In this case store an instance of the SmtpMail object in Session collection:

```
if( Session["SmtpMail"] == null )
{
    objSmtpMail = new AddEmailLib.SmtpMailClass();
    Session["SmtpMail"] = objSmtpMail;
}
else
{
    objSmtpMail = (AddEmailLib.SmtpMailClass)Session["SmtpMail"];
}
```

4. Set SMTP server address, port, username and password:

```
objSmtpMail.SmtpServer = "mail.myserver.com";
objSmtpMail.SmtpPort = 25;
objSmtpMail.SmtpUsername = "jsmith";
objSmtpMail.SmtpPassword = "mypassword";
```

5a. Send an e-mail synchronously using SimpleSend or SimpleSendAttachment:

```
string strError;
int resultCode = objSmtpMail.SimpleSend("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", out strError);
if (resultCode == 0)
{
  // E-mail was sent successfully
}
else
{
  // Send failed, strError contains detailed error description
}
```

5b. Alternatively, create and fill MailMessage object and send it synchronously using Send or asynchronously using SendAsync:

```
// Create message and setup subject and body
MailMessageClass objMessage = new AddEmailLib.MailMessageClass();
objMessage.MessageSubject = "test";
objMessage.MessageBody = "Test message";
```

```
// Setup sender
MailAddressClass objSender = new MailAddressClass();
objSender.Address = "jsmith@myserver.com";
objSender.Name = "Jane Smith";
objMessage.Sender = objSender;

// Setup first recipient
MailAddressClass objRecipient = new MailAddressClass();
objRecipient.Address = "jane@someserver.com";
objRecipient.Name = "Jane Smith";
objMessage.AddRecipient(objRecipient);

// Setup second recipient
objRecipient = new MailAddressClass();
objRecipient.Address = "james@someserver.com";
objRecipient.Name = "James Smith";
objMessage.AddRecipient(objRecipient);

// Send prepared message synchronously
string strError;
int resultCode = objSmtpMail.Send(objMessage, true, out strError);
if (resultCode == 0)
{
  // E-mail was sent successfully
}
else
{
  // Send failed, strError contains detailed error description
}

// Alternatively, send prepared message asynchronously
//int messageNumber = objSmtpMail.SendAsync(objMessage, true);
// Save the message number. It is used to retrieve the status of the message
```

6. If the message was sent asynchronously use GetStatus to check the status of the message:

```
AddEmailLib.MailStatus messageStatus = objSmtpMail.GetStatus(messageNumber);
switch(messageStatus)
{
    case AddEmailLib.MailStatus.MailStatusQueued:
    break;
    case AddEmailLib.MailStatus.MailStatusSending:
    break;
    case AddEmailLib.MailStatus.MailStatusSent:
        // E-mail was sent successfully
    break;
    case AddEmailLib.MailStatus.MailStatusFailed:
        // E-mail wasn't sent
        int errorCode = objSmtpMail.GetErrorCode(messageNumber);
        string errorDescription = objSmtpMail.GetErrorDescription(messageNumber);
    break;
    case AddEmailLib.MailStatus.MailStatusCanceled:
    break;
}
```

Please refer to the Reference section of this manual for detailed description of AddEmail objects, methods and properties. Included ASP.NET samples provide code snapshots for common operations such as sending text e-mails and sending HTML e-mails.

2.3.2 Samples

AddEmail includes the following ASP.NET / C# samples:

SimpleSend Sample

Location: *AddEmail\Samples\ASP.NET 1.x - C#\SimpleSend*
AddEmail\Samples\ASP.NET 2 - C#\SimpleSend
AddEmail\Samples\ASP.NET 4 - C#\SimpleSend

SimpleSend sample demonstrates how to send text or HTML e-mails synchronously using SimpleSend.

HtmlMail Sample

Location: *AddEmail\Samples\ASP.NET 1.x - C#\HtmlMail*
AddEmail\Samples\ASP.NET 2 - C#\HtmlMail
AddEmail\Samples\ASP.NET 4 - C#\HtmlMail

HtmlMail sample demonstrates how to send HTML e-mails synchronously using Send. This sample also shows how to set alternative text-only body of e-mail.

UnicodeMail Sample

Location: *AddEmail\Samples\ASP.NET 1.x - C#\UnicodeMail*
AddEmail\Samples\ASP.NET 2 - C#\UnicodeMail
AddEmail\Samples\ASP.NET 4 - C#\UnicodeMail

UnicodeMail sample demonstrates how to create and send e-mails that contain Unicode characters. Unicode can be used in the body of a message, in the subject field, and in the names of sender and recipients.

SendAsync Sample

Location: *AddEmail\Samples\ASP.NET 1.x - C#\SendAsync*
AddEmail\Samples\ASP.NET 2 - C#\SendAsync
AddEmail\Samples\ASP.NET 4 - C#\SendAsync

SendAsync sample demonstrates how to send HTML e-mails asynchronously using SendAsync. This sample also shows how to get status of the e-mail using GetStatus.

2.4 ASP.NET - VB

2.4.1 Using AddEmail in ASP.NET / VB projects

To use AddEmail in your ASP.NET / VB project perform the following steps:

1. Add a reference to the AddEmail library: In Visual Studio main menu, select View -> Solution Explorer to open Solution Explorer. Right-click on the project that will use AddEmail and select Add Reference from the pop-up menu. Click COM tab, find and select **AddEmail 4.0 Type Library**, click Select button then OK button.

2. Declare a variable that will hold a reference to the SmtpMail object:

```
Dim objSmtpMail As AddEmailLib.SmtpMailClass
```

3a. Create an instance of the SmtpMail object:

```
objSmtpMail = New AddEmailLib.SmtpMailClass()
```

3b. If you are sending e-mails asynchronously you will need to access the same SmtpMail object on more than one page. In this case store an instance of the SmtpMail object in Session collection:

```
If IsNothing(Session("SmtpMail")) Then
    objSmtpMail = New AddEmailLib.SmtpMailClass()
    Session("SmtpMail") = objSmtpMail
Else
    objSmtpMail = Session("SmtpMail")
End If
```

4. Set SMTP server address, port, username and password:

```
objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpPort = 25
objSmtpMail.SmtpUsername = "jsmith"
objSmtpMail.SmtpPassword = "mypassword"
```

5a. Send an e-mail synchronously using SimpleSend or SimpleSendAttachment:

```
Dim strError As String
Dim resultCode As Integer
resultCode = objSmtpMail.SimpleSend("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", strError)
If resultCode = 0 Then
  ' E-mail was sent successfully
Else
  ' Send failed, strError contains detailed error description
End If
```

5b. Alternatively, create and fill MailMessage object and send it synchronously using Send or asynchronously using SendAsync:

```
' Create message and setup subject and body
Dim objMessage As New AddEmailLib.MailMessageClass()
objMessage.MessageSubject = "test"
objMessage.MessageBody = "Test message"

' Setup sender
Dim objSender As New AddEmailLib.MailAddressClass()
objSender.Address = "jsmith@myserver.com"
objSender.Name = "John Smith"
objMessage.Sender = objSender
```

```
' Setup first recipient
Dim objRecipient As AddEmailLib.MailAddressClass
objRecipient = New AddEmailLib.MailAddressClass()
objRecipient.Address = "jane@someserver.com"
objRecipient.Name = "Jane Smith"
objMessage.AddRecipient(objRecipient)

' Setup second recipient
objRecipient = New AddEmailLib.MailAddressClass()
objRecipient.Address = "james@someserver.com"
objRecipient.Name = "James Smith"
objMessage.AddRecipient(objRecipient)

' Send prepared message synchronously
Dim strError As String
Dim resultCode As Integer
resultCode = objSmtpMail.Send(objMessage, True, strError)
If resultCode = 0 Then
  ' E-mail was sent successfully
Else
  ' Send failed, strError contains detailed error description
End If

' Alternatively, send prepared message asynchronously
'Dim messageNumber As Integer
'messageNumber = objSmtpMail.SendAsync(objMessage, True)
' Save the message number. It is used to retrieve the status of the message
```

6. If the message was sent asynchronously use GetStatus to check the status of the message:

```
Dim messageStatus As AddEmailLib.MailStatus
messageStatus = objSmtpMail.GetStatus(messageNumber)
Select Case messageStatus
    Case AddEmailLib.MailStatus.MailStatusQueued
        ...
    Case AddEmailLib.MailStatus.MailStatusSending
        ...
    Case AddEmailLib.MailStatus.MailStatusSent
        ' E-mail was sent successfully
    Case AddEmailLib.MailStatus.MailStatusFailed
        ' E-mail wasn't sent
        Dim errorCode As Integer
        errorCode = objSmtpMail.GetErrorCode(messageNumber)
        Dim errorDescription As String
        errorDescription = objSmtpMail.GetErrorDescription(messageNumber)
    Case AddEmailLib.MailStatus.MailStatusCanceled
        ...
End Select
```

Please refer to the Reference section of this manual for detailed description of AddEmail objects, methods and properties. Included ASP.NET samples provide code snapshots for common operations such as sending text e-mails and sending HTML e-mails.

2.4.2 Samples

AddEmail includes the following ASP.NET / VB samples:

SimpleSend Sample

Location: *AddEmail\Samples\ASP.NET 1.x - VB\SimpleSend*
AddEmail\Samples\ASP.NET 2 - VB\SimpleSend
AddEmail\Samples\ASP.NET 4 - VB\SimpleSend

SimpleSend sample demonstrates how to send text or HTML e-mails synchronously using SimpleSend.

HtmlMail Sample

Location: *AddEmail\Samples\ASP.NET 1.x - VB\HtmlMail*
AddEmail\Samples\ASP.NET 2 - VB\HtmlMail
AddEmail\Samples\ASP.NET 4 - VB\HtmlMail

HtmlMail sample demonstrates how to send HTML e-mails synchronously using Send. This sample also shows how to set alternative text-only body of e-mail.

UnicodeMail Sample

Location: *AddEmail\Samples\ASP.NET 1.x - VB\UnicodeMail*
AddEmail\Samples\ASP.NET 2 - VB\UnicodeMail
AddEmail\Samples\ASP.NET 4 - VB\UnicodeMail

UnicodeMail sample demonstrates how to create and send e-mails that contain Unicode characters. Unicode can be used in the body of a message, in the subject field, and in the names of sender and recipients.

SendAsync Sample

Location: *AddEmail\Samples\ASP.NET 1.x - VB\SendAsync*
AddEmail\Samples\ASP.NET 2 - VB\SendAsync
AddEmail\Samples\ASP.NET 4 - VB\SendAsync

SendAsync sample demonstrates how to send HTML e-mails asynchronously using SendAsync. This sample also shows how to get status of the e-mail using GetStatus.

2.5 VBA (Microsoft Office)

2.5.1 Using AddEmail in VBA projects

AddEmail ActiveX can be used to send emails from any Microsoft Office application that supports Visual Basic for Applications (VBA), such as Microsoft Access, Microsoft Excel, Microsoft Word etc. To use AddEmail in your VBA project perform the following steps:

1. (Optional) Add reference to AddEmail library: In VBA editor menu, select Tools -> References. Find and select **AddEmail 4.0 Type Library**.

2. Add a code that creates SmtpMail object and sends a message using SimpleSend, SimpleSendHtml, SimpleSendAttachment or Send. Please use code snapshots below to get started.

Snapshot 1: simple text email.

```
Dim objSmtpMail As Object, strError As String, numResultCode As Long
Set objSmtpMail = CreateObject("AddEmail.SmtpMail")
objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpUsername = "jsmith"
objSmtpMail.SmtpPassword = "mypassword"
numResultCode = objSmtpMail.SimpleSend("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", strError)
If numResultCode = 0 Then
    MsgBox "Sent successfully!"
Else
    MsgBox strError
End If
```

Snapshot 2: text email with attachments.

```
Dim objSmtpMail As Object, strError As String, numResultCode As Long
Set objSmtpMail = CreateObject("AddEmail.SmtpMail")
objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpUsername = "jsmith"
objSmtpMail.SmtpPassword = "mypassword"
numResultCode = objSmtpMail.SimpleSendAttachment("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", "c:\files
\doc1.pdf;c:\files\doc2.pdf", strError)
If numResultCode = 0 Then
    MsgBox "Sent successfully!"
Else
    MsgBox strError
End If
```

Snapshot 3: HTML message with attachments.

```
Dim objSmtpMail As Object, strError As String, numResultCode As Long
Set objSmtpMail = CreateObject("AddEmail.SmtpMail")
objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpUsername = "jsmith"
objSmtpMail.SmtpPassword = "mypassword"

' Create message and setup subject and body
Dim objMailMessage As Object
Set objMailMessage = CreateObject("AddEmail.MailMessage")
objMailMessage.MessageBodyFormat = 1 'HTML format
objMailMessage.MessageSubject = "test"
objMailMessage.MessageBody = "<html><body><b>Testing...</b></body></html>"

' Add first attachment
Dim objMailAttachment As Object
Set objMailAttachment = CreateObject("AddEmail.MailAttachment")
objMailAttachment.File = "c:\files\doc1.pdf"
objMailMessage.AddAttachment objMailAttachment
```

```
' Add second attachment
Set objMailAttachment = CreateObject("AddEmail.MailAttachment")
objMailAttachment.File = "c:\files\doc2.pdf"
objMailMessage.AddAttachment objMailAttachment

' Setup sender
Dim objMailAddress As Object
Set objMailAddress = CreateObject("AddEmail.MailAddress")
objMailAddress.Name = "John Smith"
objMailAddress.Address = "jsmith@myserver.com"
objMailMessage.Sender = objMailAddress

' Setup first recipient
Set objMailAddress = CreateObject("AddEmail.MailAddress")
objMailAddress.Name = "Jane Smith"
objMailAddress.Address = "jane@someserver.com"
objMailMessage.AddRecipient objMailAddress

' Setup second recipient
Set objMailAddress = CreateObject("AddEmail.MailAddress")
objMailAddress.Name = "James Smith"
objMailAddress.Address = "james@someserver.com"
objMailMessage.AddRecipient objMailAddress

' Send prepared message
numResultCode = objSmtpMail.Send(objMailMessage, True, strError)
If numResultCode = 0 Then
    MsgBox "Sent successfully!"
Else
    MsgBox strError
End If
```

Please refer to the Reference section of this manual for detailed description of AddEmail objects, methods and properties.

2.5.2 Samples

AddEmail includes the following VBA samples:

SimpleSend Sample

Location: *AddEmail\Samples\Access 2010-2016\SimpleSend.accdb*
AddEmail\Samples\Access 2000-2007\SimpleSend.adp
AddEmail\Samples\Excel 2010-2016\SimpleSend.xlsm
AddEmail\Samples\Excel 2000-2007\SimpleSend.xls

SimpleSend sample demonstrates how to send text or HTML e-mails synchronously using SimpleSend.

HtmlMail Sample

Location: *AddEmail\Samples\Access 2010-2016\HtmlMail.accdb*
AddEmail\Samples\Access 2000-2007\HtmlMail.adp

HtmlMail sample demonstrates how to send HTML e-mails with attachments using Send. This sample also shows how to set alternative text-only body of the e-mail.

SendAsync Sample

Location: *AddEmail\Samples\Access 2010-2016\SendAsync.accdb*
AddEmail\Samples\Access 2000-2007\SendAsync.adp

SendAsync sample demonstrates how to send HTML e-mails with attachments asynchronously using SendAsync. This sample also shows how to process events from SmtpMail object.

2.6 C++/MFC/ATL

2.6.1 Using AddEmail in C++ projects

To use AddEmail in your C++ project perform the following steps:

1. Import AddEmail type library: add #import directive to StdAfx.h or other appropriate header.

```
#import "AddEmail\AddEmail.tlb" named_guids
```

2. Declare a variable that will hold a reference to the SmtpMail object:

```
private AddEmailLib::ISmtpMailPtr m_spSmtpMail;
```

3. Create an instance of the SmtpMail object:

```
m_spSmtpMail.CreateInstance(__uuidof(AddEmailLib::SmtpMail));
```

4. Set SMTP server address, port, username and password:

```
m_spSmtpMail->PutSmtpServer("mail.myserver.com");
m_spSmtpMail->PutSmtpPort(25);
m_spSmtpMail->PutSmtpUsername("jsmith");
m_spSmtpMail->PutSmtpPassword("mypassword");
```

5a. Send an e-mail synchronously using SimpleSend or SimpleSendAttachment:

```
BSTR bstrError;
int resultCode = m_spSmtpMail->SimpleSend("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", &bstrError);
if (resultCode == 0)
{
  // E-mail was sent successfully
}
else
{
  // Send failed, bstrError contains detailed error description
}
```

5b. Alternatively, create and fill MailMessage object and send it synchronously using Send or asynchronously using SendAsync:

```
// Create message and setup subject and body
AddEmailLib::IMailMessagePtr spMessage;
spMessage.CreateInstance(__uuidof(AddEmailLib::MailMessage));
spMessage->PutMessageSubject("test");
spMessage->PutMessageBody("Test message");

// Setup sender
AddEmailLib::IMailAddressPtr spFrom;
spFrom.CreateInstance( __uuidof(AddEmailLib::MailAddress) );
spFrom->PutAddress("jsmith@myserver.com");
spFrom->PutName("John Smith");
spMessage->PutSender(spFrom);

// Setup first recipient
AddEmailLib::IMailAddressPtr spTo;
spTo.CreateInstance(__uuidof(AddEmailLib::MailAddress));
spTo->PutAddress("jane@someserver.com");
spTo->PutName("Jane Smith");
spMessage->AddRecipient(spTo);

// Setup second recipient
spTo.CreateInstance(__uuidof(AddEmailLib::MailAddress));
spTo->PutAddress("james@someserver.com");
spTo->PutName("James Smith");
spMessage->AddRecipient(spTo);

// Send prepared message
BSTR bstrError;
int resultCode = m_spSmtpMail->Send(spMessage, TRUE, &bstrError);
if (resultCode == 0)
{
  // E-mail was sent successfully
}
else
{
  // Send failed, bstrError contains detailed error description
}

// Alternatively, send prepared message asynchronously
//int messageNumber = m_spSmtpMail->SendAsync(spMessage, TRUE);
```

6. If the e-mail is sent asynchronously your application will need to process events to find out when the email was sent successfully or send operation failed. To process events from the SmtpMail object you need to add ATL support to your project, inherit your class from IDispEventImpl<> template, declare event handlers, create sink map and subscribe to events:

```
class CMyClass :
    public IDispEventImpl<1, CMyClass, &AddEmailLib::DIID__ISmtpMailEvents,
                          &AddEmailLib::LIBID_AddEmailLib, 1, 0>
{
public:
    // SmtpMail events
    void __stdcall OnStatusChange(LONG messageNumber, AddEmailLib::MailStatus
```

```
newStatus);
    void __stdcall OnProgress(LONG messageNumber, LONG bytesSent, LONG bytesTotal);

    // ATL sink for SmtpMail events
    BEGIN_SINK_MAP(CMyClass)
            SINK_ENTRY_EX(1, AddEmailLib::DIID__ISmtpMailEvents, 1, OnStatusChange)
            SINK_ENTRY_EX(1, AddEmailLib::DIID__ISmtpMailEvents, 2, OnProgress)
    END_SINK_MAP()
}

// Subscribe to events
DispEventAdvise(m_spSmtpMail);
```

Please refer to the Reference section of this manual for detailed description of AddEmail objects, methods and properties. Included C++ samples provide code snapshots for common operations such as sending text e-mails, sending HTML e-mails, adding attachments to e-mails, creating HTML e-mails with embedded images.

2.6.2 Samples

AddEmail includes the following C++ samples:

SimpleSend Sample

Location: *AddEmail**Samples**C++**SimpleSend*\

SimpleSend sample demonstrates how to send text or HTML e-mails using SimpleSend.

SimpleSendAttachment Sample

Location: *AddEmail**Samples**C++**SimpleSendAttachment*\

SimpleSendAttachment sample demonstrates how to send text or HTML e-mails with attachments using SimpleSendAttachment.

HtmlMail Sample

Location: *AddEmail**Samples**C++**HtmlMail*\

HtmlMail sample demonstrates how to send HTML e-mails with attachments using Send. This sample also shows how to set alternative text-only body of e-mail.

EmbeddedImages Sample

Location: *AddEmail**Samples**C++**EmbeddedImages*\

EmbeddedImages sample demonstrates how to create and send HTML e-mails with embedded images using Send.

EnterpriseMail Sample

Location: *AddEmail**Samples**C++**EnterpriseMail*\

EnterpriseMail sample demonstrates features of AddEmail Enterprise version. It shows how to queue and send multiple e-mails simultaneously using AddEmail multi-threading support. This sample also shows how to process events from SmtpMail object. In addition, this sample shows how to send e-mails directly to recipients' mail servers without using outgoing mail server of your organization or internet provider.

UnicodeMail Sample

Location: *AddEmail**Samples**C++**UnicodeMail*\

UnicodeMail sample demonstrates how to create and send e-mails that contain Unicode characters. Unicode can be used in the body of a message, in the subject field, and in the names of sender and recipients.

SendAsync Sample

Location: *AddEmail**Samples**C++**SendAsync*\

HtmlMail sample demonstrates how to send HTML e-mails with attachments asynchronously using SendAsync. This sample also shows how to process events from SmtpMail object.

2.7 VB6

2.7.1 Using AddEmail in VB6 projects

To use AddEmail in your Visual Basic 6 project perform the following steps:

1. Add a reference to the AddEmail library: In VB6 main menu, select Project -> References. Find and select **AddEmail 4.0 Type Library**.

2. Declare a variable that will hold a reference to the SmtpMail object:

```
Dim WithEvents objSmtpMail As AddEmailLib.SmtpMail
```

3. Create an instance of SmtpMail object in Form_Load() or other functions:

```
Set objSmtpMail = New AddEmailLib.SmtpMail
```

4. Set SMTP server address, port, username and password:

```
objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpPort = 25
objSmtpMail.SmtpUsername = "jsmith"
objSmtpMail.SmtpPassword = "mypassword"
```

5a. Send an e-mail synchronously using SimpleSend or SimpleSendAttachment:

```
Dim strError As String
Dim resultCode As Long
resultCode = objSmtpMail.SimpleSend("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", strError)
If resultCode = 0 Then
```

```
  ' E-mail was sent successfully
Else
  ' Send failed, strError contains detailed error description
End If
```

5b. Alternatively, create and fill MailMessage object and send it synchronously using Send or asynchronously using SendAsync:

```
' Create message and setup subject and body
Dim objMessage As New AddEmailLib.MailMessage
objMessage.MessageSubject = "test"
objMessage.MessageBody = "Test message"

' Setup sender
Dim objSender As New AddEmailLib.MailAddress
objSender.Address = "jsmith@myserver.com"
objSender.Name = "John Smith"
objMessage.Sender = objSender

' Setup first recipient
Dim objRecipient As AddEmailLib.MailAddress
Set objRecipient = New AddEmailLib.MailAddress
objRecipient.Address = "jane@someserver.com"
objRecipient.Name = "Jane Smith"
objMessage.AddRecipient objRecipient

' Setup second recipient
Set objRecipient = New AddEmailLib.MailAddress
objRecipient.Address = "james@someserver.com"
objRecipient.Name = "James Smith"
objMessage.AddRecipient objRecipient

' Send prepared message synchronously
Dim strError As String
Dim resultCode As Long
resultCode = objSmtpMail.Send(objMessage, True, strError)
If resultCode = 0 Then
  ' E-mail was sent successfully
Else
  ' Send failed, strError contains detailed error description
End If

' Alternatively, send prepared message asynchronously
'Dim messageNumber As Long
'messageNumber = objSmtpMail.SendAsync(objMessage, True)
```

6. If the e-mail is sent asynchronously your application will need to process events to find out when the email was sent successfully or send operation failed. To process events from the SmtpMail object declare event handlers as shown below:

```
Private Sub objSmtpMail_OnStatusChange(ByVal messageNumber As Long, ByVal newStatus As
Long)
    ' Event processing code
End Sub

Private Sub objSmtpMail_OnProgress(ByVal messageNumber As Long, ByVal bytesSent As
```

```
Long, ByVal bytesTotal As Long)
    ' Event processing code
End Sub
```

Please refer to the Reference section of this manual for detailed description of AddEmail objects, methods and properties. Included Visual Basic samples provide code snapshots for common operations such as sending text e-mails, sending HTML e-mails, adding attachments to e-mails, creating HTML e-mails with embedded images.

2.7.2 Samples

AddEmail includes the following Visual Basic 6 samples:

SimpleSend Sample

Location: *AddEmail\Samples\VB6\SimpleSend*

SimpleSend sample demonstrates how to send text or HTML e-mails synchronously using SimpleSend.

SimpleSendAttachment Sample

Location: *AddEmail\Samples\VB6\SimpleSendAttachment*

SimpleSendAttachment sample demonstrates how to send text or HTML e-mails with attachments synchronously using SimpleSendAttachment.

HtmlMail Sample

Location: *AddEmail\Samples\VB6\HtmlMail*

HtmlMail sample demonstrates how to send HTML e-mails with attachments using Send. This sample also shows how to set alternative text-only body of e-mail.

ImportHtmlWithImages Sample

Location: *AddEmail\Samples\VB6\ImportHtmlWithImages*

ImportHtmlWithImages sample demonstrates how to import HTML file with embedded images using ImportHTML.

EmbeddedImages Sample

Location: *AddEmail\Samples\VB6\EmbeddedImages*

EmbeddedImages sample demonstrates how to create and send HTML e-mails with embedded images using Send.

EnterpriseMail Sample

Location: *AddEmail\Samples\VB6\EnterpriseMail*

EnterpriseMail sample demonstrates features of AddEmail Enterprise version. It shows how to queue and send multiple e-mails simultaneously using AddEmail multi-threading support. This sample also shows how to process events from SmtpMail object. In addition, this sample also shows how to send e-mails directly to recipients' mail servers without using outgoing mail server of your organization or internet provider.

SendAsync Sample

Location: *AddEmail\Samples\VB6\SendAsync*

HtmlMail sample demonstrates how to send HTML e-mails with attachments asynchronously using SendAsync. This sample also shows how to process events from SmtpMail object.

2.8 VBScript

2.8.1 Using AddEmail in VBScript projects

AddEmail ActiveX can be used with any environment that supports VBScript. Windows has built-in VBScript support and can execute VBScript files that have .VBS extension. To use AddEmail ActiveX from VBScript code create SmtpMail object and send a message using SimpleSendScriptable, SimpleSendAttachmentScriptable or SendScriptable. Please use code snapshots below to get started.

Snapshot 1: simple text email.

```
Dim objSmtpMail, strError, numResultCode
Set objSmtpMail = CreateObject("AddEmail.SmtpMail")
objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpUsername = "jsmith"
objSmtpMail.SmtpPassword = "mypassword"
numResultCode = objSmtpMail.SimpleSendScriptable("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", strError)
If numResultCode = 0 Then
    MsgBox "Sent successfully!"
Else
    MsgBox strError
End If
```

Snapshot 2: text email with attachments.

```
Dim objSmtpMail, strError, numResultCode
Set objSmtpMail = CreateObject("AddEmail.SmtpMail")
objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpUsername = "jsmith"
objSmtpMail.SmtpPassword = "mypassword"
numResultCode = objSmtpMail.SimpleSendAttachmentScriptable("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", "c:\files
\doc1.pdf;c:\files\doc2.pdf", strError)
If numResultCode = 0 Then
    MsgBox "Sent successfully!"
Else
    MsgBox strError
End If
```

Snapshot 3: HTML message with attachments.

```
Dim objSmtpMail, objMailMessage, objMailAttachment, objMailAddress, strError,
numResultCode
Set objSmtpMail = CreateObject("AddEmail.SmtpMail")
objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpUsername = "jsmith"
objSmtpMail.SmtpPassword = "mypassword"

' Create message and setup subject and body
Set objMailMessage = CreateObject("AddEmail.MailMessage")
objMailMessage.MessageBodyFormat = 1 'HTML format
objMailMessage.MessageSubject = "test"
objMailMessage.MessageBody = "<html><body><b>Testing...</b></body></html>"

' Add first attachment
Set objMailAttachment = CreateObject("AddEmail.MailAttachment")
objMailAttachment.File = "c:\files\doc1.pdf"
objMailMessage.AddAttachment(objMailAttachment)

' Add second attachment
Set objMailAttachment = CreateObject("AddEmail.MailAttachment")
objMailAttachment.File = "c:\files\doc2.pdf"
objMailMessage.AddAttachment(objMailAttachment)

' Setup sender
Set objMailAddress = CreateObject("AddEmail.MailAddress")
objMailAddress.Name = "John Smith"
objMailAddress.Address = "jsmith@myserver.com"
objMailMessage.Sender = objMailAddress

' Setup first recipient
Set objMailAddress = CreateObject("AddEmail.MailAddress")
objMailAddress.Name = "Jane Smith"
objMailAddress.Address = "jane@someserver.com"
objMailMessage.AddRecipient(objMailAddress)

' Setup second recipient
Set objMailAddress = CreateObject("AddEmail.MailAddress")
objMailAddress.Name = "James Smith"
objMailAddress.Address = "james@someserver.com"
objMailMessage.AddRecipient(objMailAddress)

' Send prepared message
numResultCode = objSmtpMail.SendScriptable(objMailMessage, True, strError)
If numResultCode = 0 Then
    MsgBox "Sent successfully!"
Else
    MsgBox strError
End If
```

Please refer to the Reference section of this manual for detailed description of AddEmail objects, methods and properties. Included VBScript samples provide code snapshots for common operations such as sending text e-mails, sending HTML e-mails, adding attachments to e-mails.

2.8.2 Samples

AddEmail includes the following VBScript samples:

SimpleSend Sample

Location: *AddEmail\Samples\VBS\SimpleSend.vbs*

SimpleSend sample demonstrates how to send text or HTML e-mails synchronously using SimpleSendScriptable.

SimpleSendAttachment Sample

Location: *AddEmail\Samples\VBS\SimpleSendAttachment.vbs*

SimpleSendAttachment sample demonstrates how to send text or HTML e-mails with attachments synchronously using SimpleSendAttachmentScriptable.

HtmlMail Sample

Location: *AddEmail\Samples\VBS\HtmlMail.vbs*

HtmlMail sample demonstrates how to send HTML e-mails with attachments synchronously using SendScriptable. This sample also shows how to set alternative text-only body of e-mail.

ImportHtmlWithImages Sample

Location: *AddEmail\Samples\VBS\ImportHtmlWithImages*

ImportHtmlWithImages sample demonstrates how to import HTML file with embedded images using ImportHTML.

EmbeddedImages Sample

Location: *AddEmail\Samples\VBS\EmbeddedImages*

EmbeddedImages sample demonstrates how to create and send HTML e-mails with embedded images using Send.

2.9 JScript

2.9.1 Using AddEmail in JScript projects

AddEmail ActiveX can be used with any environment that supports JScript. Windows has built-in JScript support and can execute JScript files that have .JS extension. To use AddEmail ActiveX from JScript code create SmtpMail object and send a message using SimpleSendScriptable, SimpleSendAttachmentScriptable or SendScriptable. Please use code snapshots below to get started.

Snapshot 1: simple text email.

```
var objSmtpMail, strError, numResultCode;
objSmtpMail = new ActiveXObject("AddEmail.SmtpMail");
objSmtpMail.SmtpServer = "mail.myserver.com";
objSmtpMail.SmtpUsername = "jsmith";
objSmtpMail.SmtpPassword = "mypassword";
numResultCode = objSmtpMail.SimpleSendScriptable("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", strError);
if (numResultCode == 0)
{
    WScript.Echo("Sent successfully!");
}
else
{
    strError = objSmtpMail.GetLastErrorDescription();
    WScript.Echo(strError);
}
```

Snapshot 2: text email with attachments.

```
var objSmtpMail, strError, numResultCode;
objSmtpMail = new ActiveXObject("AddEmail.SmtpMail")
objSmtpMail.SmtpServer = "mail.myserver.com";
objSmtpMail.SmtpUsername = "jsmith";
objSmtpMail.SmtpPassword = "mypassword";
numResultCode = objSmtpMail.SimpleSendAttachmentScriptable("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", "c:\\files\
\doc1.pdf;c:\\files\\doc2.pdf", strError);
if (numResultCode == 0)
{
    WScript.Echo("Sent successfully!");
}
else
{
    strError = objSmtpMail.GetLastErrorDescription();
    WScript.Echo(strError);
}
```

Snapshot 3: HTML message with attachments.

```
var objSmtpMail, objMailMessage, objMailAttachment, objMailAddress, strError,
numResultCode
objSmtpMail = new ActiveXObject("AddEmail.SmtpMail");
objSmtpMail.SmtpServer = "mail.myserver.com";
objSmtpMail.SmtpUsername = "jsmith";
objSmtpMail.SmtpPassword = "mypassword";
objSmtpMail.SmtpSSL = false;

// Create message and setup subject and body
objMailMessage = new ActiveXObject("AddEmail.MailMessage");
objMailMessage.MessageBodyFormat = 1; //HTML format
objMailMessage.MessageSubject = "test";
objMailMessage.MessageBody = "<html><body><b>Testing...</b></body></html>";

// Add first attachment
objMailAttachment = new ActiveXObject("AddEmail.MailAttachment");
objMailAttachment.File = "c:\\files\\doc1.pdf";
```

```
objMailMessage.AddAttachment(objMailAttachment);

// Add second attachment
objMailAttachment = new ActiveXObject("AddEmail.MailAttachment");
objMailAttachment.File = "c:\\files\\doc2.pdf";
objMailMessage.AddAttachment(objMailAttachment);

// Setup sender
objMailAddress = new ActiveXObject("AddEmail.MailAddress");
objMailAddress.Name = "John Smith";
objMailAddress.Address = "jsmith@myserver.com";
objMailMessage.Sender = objMailAddress;

// Setup first recipient
objMailAddress = new ActiveXObject("AddEmail.MailAddress");
objMailAddress.Name = "Jane Smith";
objMailAddress.Address = "jane@someserver.com";
objMailMessage.AddRecipient(objMailAddress);

// Setup second recipient
objMailAddress = new ActiveXObject("AddEmail.MailAddress");
objMailAddress.Name = "James Smith";
objMailAddress.Address = "james@someserver.com";
objMailMessage.AddRecipient(objMailAddress);

// Send prepared message synchronously
numResultCode = objSmtpMail.SendScriptable(objMailMessage, true, strError);
if (numResultCode == 0)
{
    WScript.Echo("Sent successfully!");
}
else
{
    strError = objSmtpMail.GetLastErrorDescription();
    WScript.Echo(strError);
}
```

Please refer to the Reference section of this manual for detailed description of AddEmail objects, methods and properties. Included JScript samples provide code snapshots for common operations such as sending text e-mails, sending HTML e-mails, adding attachments to e-mails.

2.9.2 Samples

AddEmail includes the following JScript samples:

SimpleSend Sample

Location: *AddEmail\Samples\JS\SimpleSend.js*

SimpleSend sample demonstrates how to send text or HTML e-mails synchronously using SimpleSendScriptable.

SimpleSendAttachment Sample

Location: *AddEmail\Samples\JS\SimpleSendAttachment.js*

SimpleSendAttachment sample demonstrates how to send text or HTML e-mails with attachments synchronously using SimpleSendAttachmentScriptable.

HtmlMail Sample

Location: *AddEmail\Samples\JS\HtmlMail.js*

HtmlMail sample demonstrates how to send HTML e-mails with attachments synchronously using SendScriptable. This sample also shows how to set alternative text-only body of e-mail.

ImportHtmlWithImages Sample

Location: *AddEmail\Samples\VBS\ImportHtmlWithImages*

ImportHtmlWithImages sample demonstrates how to import HTML file with embedded images using ImportHTML.

EmbeddedImages Sample

Location: *AddEmail\Samples\VBS\EmbeddedImages*

EmbeddedImages sample demonstrates how to create and send HTML e-mails with embedded images using Send.

2.10 ASP - VBScript

2.10.1 Using AddEmail in ASP / VBScript projects

To use AddEmail in your ASP / VBScript project perform the following steps:

1. (Optional) If you are using Visual Interdev add a reference to the AddEmail library: In Visual Interdev main menu, select Project -> References. Find and select **AddEmail 4.0 Type Library**.

2. Declare variable that will hold a reference to the SmtpMail object:

```
Dim objSmtpMail
```

3a. Create an instance of the SmtpMail object:

```
Set objSmtpMail = Server.CreateObject("AddEmail.SmtpMail")
```

3b. If you are sending e-mails asynchronously you will need to access the same SmtpMail object on more than one page. In this case store an instance of the SmtpMail object in Session collection:

```
If IsEmpty(Session("SmtpMail")) Then
    Set objSmtpMail = Server.CreateObject("AddEmail.SmtpMail")
    Set Session("SmtpMail") = objSmtpMail
```

```
Else
    Set objSmtpMail = Session("SmtpMail")
End If
```

4. Set SMTP server address, port, username and password:

```
objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpPort = 25
objSmtpMail.SmtpUsername = "jsmith"
objSmtpMail.SmtpPassword = "mypassword"
```

5a. Send an e-mail synchronously using SimpleSendScriptable or SimpleSendAttachmentScriptable:

```
Dim strError, numResultCode
numResultCode = objSmtpMail.SimpleSendScriptable("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", strError)
If numResultCode = 0 Then
    ' The email was sent successfully
    Response.Write "Sent successfully!"
Else
    ' The email was not sent, strError contains error description
    Response.Write strError
End If
```

5b. Alternatively, create and fill MailMessage object and send it synchronously using SendScriptable or asynchronously using SendAsync:

```
Dim objMailMessage, objMailAttachment, objMailAddress, strError, numResultCode

' Create message and setup subject and body
Set objMailMessage = CreateObject("AddEmail.MailMessage")
objMailMessage.MessageBodyFormat = 1 'HTML format
objMailMessage.MessageSubject = "test"
objMailMessage.MessageBody = "<html><body><b>Testing...</b></body></html>"

' Add first attachment
Set objMailAttachment = CreateObject("AddEmail.MailAttachment")
objMailAttachment.File = "c:\files\doc1.pdf"
objMailMessage.AddAttachment(objMailAttachment)

' Add second attachment
Set objMailAttachment = CreateObject("AddEmail.MailAttachment")
objMailAttachment.File = "c:\files\doc2.pdf"
objMailMessage.AddAttachment(objMailAttachment)

' Setup sender
Set objMailAddress = CreateObject("AddEmail.MailAddress")
objMailAddress.Name = "John Smith"
objMailAddress.Address = "jsmith@myserver.com"
objMailMessage.Sender = objMailAddress

' Setup first recipient
Set objMailAddress = CreateObject("AddEmail.MailAddress")
objMailAddress.Name = "Jane Smith"
objMailAddress.Address = "jane@someserver.com"
objMailMessage.AddRecipient(objMailAddress)
```

```
' Setup second recipient
Set objMailAddress = CreateObject("AddEmail.MailAddress")
objMailAddress.Name = "James Smith"
objMailAddress.Address = "james@someserver.com"
objMailMessage.AddRecipient(objMailAddress)

' Send prepared message synchronously
Dim strError, numResultCode
numResultCode = objSmtpMail.SendScriptable(objMailMessage, True, strError)
If numResultCode = 0 Then
    ' The email was sent successfully
    Response.Write "Sent successfully!"
Else
    ' The email was not sent, strError contains error description
    Response.Write strError
End If

' Alternatively, send prepared message asynchronously
'Dim messageNumber
'messageNumber = objSmtpMail.SendAsync(objMailMessage, True)
' Save the message number. It is used to retrieve the status of the message
```

6. If the message was sent asynchronously use GetStatus to check the status of the message:

```
Dim numStatus
numStatus = objSmtpMail.GetStatus(numMessageNumber)
Select Case numStatus
    Case 0 'MailStatusQueued
        Response.Write "Queued"
    Case 1 'MailStatusSending
        Response.Write "Sending..."
    Case 2 'MailStatusSent
        Response.Write "Sent successfully!"
    Case 3 'MailStatusFailed
        Response.Write "Error " & objSmtpMail.GetErrorCode(numMessageNumber)
        Response.Write "<BR>" & objSmtpMail.GetErrorDescription(numMessageNumber)
    Case 4 'MailStatusCanceled
        Response.Write "Message canceled"
End Select
```

Please refer to the Reference section of this manual for detailed description of AddEmail objects, methods and properties. Included ASP samples provide code snapshots for common operations such as sending text e-mails and sending HTML e-mails.

2.10.2 Samples

AddEmail includes the following ASP / VBScript samples:

SimpleSend Sample

Location: *AddEmail\Samples\ASP - VBScript\SimpleSend*

SimpleSend sample demonstrates how to send text ot HTML e-mails synchronously using SimpleSend method.

HtmlMail Sample

Location: *AddEmail\Samples\ASP - VBScript\HtmlMail*

HtmlMail sample demonstrates how to send HTML e-mails synchronously using SendScriptable method. This sample also shows how to set alternative text-only body of e-mail.

UnicodeMail Sample

Location: *AddEmail\Samples\ASP - VBScript\UnicodeMail*

UnicodeMail sample demonstrates how to create and send e-mails that contain Unicode characters. Unicode can be used in the body of a message, in the subject field, and in the names of sender and recipients.

SendAsync Sample

Location: *AddEmail\Samples\ASP - VBScript\SendAsync*

SendAsync sample demonstrates how to send HTML e-mails asynchronously using SendAsync method. This sample also shows how to get status of the e-mail using GetStatus.

2.11 ASP - JScript

2.11.1 Using AddEmail in ASP / JScript projects

To use AddEmail in your ASP / JScript project perform the following steps:

1. (Optional) If you are using Visual Interdev add a reference to the AddEmail library: In Visual Interdev main menu, select Project -> References. Find and select **AddEmail 4.0 Type Library**.

2. Declare variable that will hold a reference to the SmtpMail object:

```
var objSmtpMail;
```

3a. Create an instance of the SmtpMail object:

```
objSmtpMail = Server.CreateObject("AddEmail.SmtpMail");
```

3b. If you are sending e-mails asynchronously you will need to access the same SmtpMail object on more than one page. In this case store an instance of the SmtpMail object in Session collection:

```
if (Session("SmtpMail") == null)
{
    objSmtpMail = Server.CreateObject("AddEmail.SmtpMail");
    Session("SmtpMail") = objSmtpMail;
}
else
{
```

```
    objSmtpMail = Session("SmtpMail");
}
```

4. Set SMTP server address, port, username and password:

```
objSmtpMail.SmtpServer = "mail.myserver.com";
objSmtpMail.SmtpPort = 25;
objSmtpMail.SmtpUsername = "jsmith";
objSmtpMail.SmtpPassword = "mypassword";
```

5a. Send an e-mail synchronously using SimpleSendScriptable or SimpleSendAttachmentScriptable:

```
var strError, numResultCode;
numResultCode = objSmtpMail.SimpleSendScriptable("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", strError);
if (numResultCode == 0)
{
    // The email was sent successfully
    Response.Write("Sent successfully!");
}
else
{
    // The email was not sent, strError contains error description
    strError = objSmtpMail.GetLastErrorDescription();
    Response.Write(strError);
}
```

5b. Alternatively, create and fill MailMessage object and send it synchronously using SendScriptable or asynchronously using SendAsync:

```
var objMailMessage, objMailAttachment, objMailAddress, strError, numResultCode;

// Create message and setup subject and body
objMailMessage = new ActiveXObject("AddEmail.MailMessage");
objMailMessage.MessageBodyFormat = 1; //HTML format
objMailMessage.MessageSubject = "test";
objMailMessage.MessageBody = "<html><body><b>Testing...</b></body></html>";

// Add first attachment
objMailAttachment = new ActiveXObject("AddEmail.MailAttachment");
objMailAttachment.File = "c:\\files\\doc1.pdf";
objMailMessage.AddAttachment(objMailAttachment);

// Add second attachment
objMailAttachment = new ActiveXObject("AddEmail.MailAttachment");
objMailAttachment.File = "c:\\files\\doc2.pdf";
objMailMessage.AddAttachment(objMailAttachment);

// Setup sender
objMailAddress = new ActiveXObject("AddEmail.MailAddress");
objMailAddress.Name = "John Smith";
objMailAddress.Address = "jsmith@myserver.com";
objMailMessage.Sender = objMailAddress;

// Setup first recipient
objMailAddress = new ActiveXObject("AddEmail.MailAddress");
```

```
objMailAddress.Name = "Jane Smith";
objMailAddress.Address = "jane@someserver.com";
objMailMessage.AddRecipient(objMailAddress);

// Setup second recipient
objMailAddress = new ActiveXObject("AddEmail.MailAddress");
objMailAddress.Name = "James Smith";
objMailAddress.Address = "james@someserver.com";
objMailMessage.AddRecipient(objMailAddress);

// Send prepared message synchronously
numResultCode = objSmtpMail.SendScriptable(objMailMessage, true, strError);
if (numResultCode == 0)
{
    // The email was sent successfully
    Response.Write("Sent successfully!");
}
else
{
    // The email was not sent, strError contains error description
    strError = objSmtpMail.GetLastErrorDescription();
    Response.Write(strError);
}

// Alternatively, send prepared message asynchronously
// var messageNumber;
// messageNumber = objSmtpMail.SendAsync(objMailMessage, true);
// Save the message number. It is used to retrieve the status of the message
```

6. If the message was sent asynchronously use GetStatus to check the status of the message:

```
var numStatus;
numStatus = objSmtpMail.GetStatus(numMessageNumber);
switch(numStatus)
{
    case 0: //MailStatusQueued
        Response.Write("Queued");
    break;
    case 1: //MailStatusSending
        Response.Write("Sending...");
    break;
    case 2: //MailStatusSent
        Response.Write("Sent successfully!");
    break;
    case 3: //MailStatusFailed
        Response.Write("Error ");
        Response.Write(objSmtpMail.GetErrorCode(numMessageNumber));
        Response.Write("<BR>");
        Response.Write(objSmtpMail.GetErrorDescription(numMessageNumber));
    break;
    case 4: //MailStatusCanceled
        Response.Write "Message canceled";
    break;
}
```

Please refer to the Reference section of this manual for detailed description of AddEmail objects,

methods and properties. Included ASP samples provide code snapshots for common operations such as sending text e-mails and sending HTML e-mails.

2.11.2 Samples

AddEmail includes the following ASP / JScript samples:

SimpleSend Sample

Location: *AddEmail\Samples\ASP - JScript\SimpleSend*

SimpleSend sample demonstrates how to send text ot HTML e-mails synchronously using SimpleSend.

HtmlMail Sample

Location: *AddEmail\Samples\ASP - JScript\HtmlMail*

HtmlMail sample demonstrates how to send HTML e-mails synchronously using SendScriptable. This sample also shows how to set alternative text-only body of e-mail.

UnicodeMail Sample

Location: *AddEmail\Samples\ASP - JScript\UnicodeMail*

UnicodeMail sample demonstrates how to create and send e-mails that contain Unicode characters. Unicode can be used in the body of a message, in the subject field, and in the names of sender and recipients.

SendAsync Sample

Location: *AddEmail\Samples\ASP - JScript\SendAsync*

SendAsync sample demonstrates how to send HTML e-mails asynchronously using SendAsync method. This sample also shows how to get status of the e-mail using GetStatus.

2.12 Delphi

2.12.1 Using AddEmail in Delphi projects

To use AddEmail in your Delphi project perform the following steps:

1. Import AddEmail ActiveX control.

If you have **Delphi 2005** or later: In Delphi main menu, select File -> New -> Package - Delphi Win32. Save the package as AddEmail_Package using File -> Save As menu. From Delphi main menu select Component -> Import Component. Select 'Import ActiveX Control' and click Next. Find and select **AddEmail 4.0 Type Library**, then click Add button. Click Next, make sure Palette Page is set to ActiveX and click Next. Select 'Add unit to AddEmail_Package project' and click Finish. In the Project Manager pane, right-click on AddEmail_Package and select Install. Delphi should display a message confirming that the package is installed. After that AddEmail will be installed to the ActiveX tab of

Components palette.

If you have older version of Delphi: In Delphi main menu, select Components -> Import ActiveX Control. Find and select **AddEmail 4.0 Type Library**, then click Install button. AddEmail ActiveX will be installed to the ActiveX tab of Components palette.

2. Place SmtpMail object of AddEmail control on the form: Select ActiveX tab of Components palette, click SmtpMail and drag a rectangle on the form. AddEmail control is not visible on runtime, so you can place it anywhere on the form.

3. To process events from SmtpMail object, create event handlers: Click SmtpMail on the form and select Events tab in the Object Inspector window. You will see 2 events OnProgress and OnStatusChange. Double-click dropdown box next to event name to create event handler.

Please refer to the Reference section of this manual for detailed description of AddEmail objects, methods and properties. Included Delphi samples provide code snapshots for common operations such as sending text e-mails, sending HTML e-mails, adding attachments to e-mails and processing events.

2.12.2 Samples

AddEmail includes the following Delphi samples:

SimpleSend Sample

Location: *AddEmail\Samples\Delphi 7\SimpleSend*

SimpleSend sample demonstrates how to send text or HTML e-mails synchronously using SimpleSend.

SimpleSendAttachment Sample

Location: *AddEmail\Samples\Delphi 7\SimpleSendAttachment*

SimpleSendAttachment sample demonstrates how to send text or HTML e-mails with attachments synchronously using SimpleSendAttachment.

HtmlMail Sample

Location: *AddEmail\Samples\Delphi 7\HtmlMail*

HtmlMail sample demonstrates how to send HTML e-mails with attachments synchronously using Send method. This sample also shows how to set alternative text-only body of e-mail.

SendAsync Sample

Location: *AddEmail\Samples\Delphi 7\SendAsync*

SendAsync sample demonstrates how to send HTML e-mails with attachments asynchronously using SendAsync method. This sample also shows how to process events from SmtpMail object.

2.13 Visual FoxPro

2.13.1 Using AddEmail in FoxPro projects

AddEmail ActiveX can be used to send emails from Visual FoxPro applications. To use AddEmail in your FoxPro project you need to add a code that creates SmtpMail object and sends a message using SimpleSend, SimpleSendAttachment or Send. Please use code snapshots below to get started.

Snapshot 1: simple text email.

```
PRIVATE objSmtpMail, strError, numResultCode
objSmtpMail = CREATEOBJECT("AddEmail.SmtpMail")
objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpUsername = "jsmith"
objSmtpMail.SmtpPassword = "mypassword"
strError = ""
numResultCode = objSmtpMail.SimpleSend("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", @strError)
IF numResultCode = 0 THEN
    MESSAGEBOX("Sent successfully!")
ELSE
    MESSAGEBOX(strError)
ENDIF
```

Snapshot 2: text email with attachments.

```
PRIVATE objSmtpMail, strError, numResultCode
objSmtpMail = CREATEOBJECT("AddEmail.SmtpMail")
objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpUsername = "jsmith"
objSmtpMail.SmtpPassword = "mypassword"
strError = ""
numResultCode = objSmtpMail.SimpleSendAttachment("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", "c:\files
\doc1.pdf;c:\files\doc2.pdf", @strError)
IF numResultCode = 0 THEN
    MESSAGEBOX("Sent successfully!")
ELSE
    MESSAGEBOX(strError)
ENDIF
```

Snapshot 3: HTML message with attachments.

```
PRIVATE objSmtpMail, objMailMessage, objMailAttachment, objMailAddress
PRIVATE strError, numResultCode
objSmtpMail = CREATEOBJECT("AddEmail.SmtpMail")
objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpUsername = "jsmith"
objSmtpMail.SmtpPassword = "mypassword"

* Create message and setup subject and body
objMailMessage = CREATEOBJECT("AddEmail.MailMessage")
objMailMessage.MessageBodyFormat = 1 && HTML format
```

```
objMailMessage.MessageSubject = "test"
objMailMessage.MessageBody = "<html><body><b>Testing...</b></body></html>"

* Add first attachment
objMailAttachment = CREATEOBJECT("AddEmail.MailAttachment")
objMailAttachment.File = "c:\files\doc1.pdf"
objMailMessage.AddAttachment(objMailAttachment)

* Add second attachment
objMailAttachment = CREATEOBJECT("AddEmail.MailAttachment")
objMailAttachment.File = "c:\files\doc2.pdf"
objMailMessage.AddAttachment(objMailAttachment)

* Setup sender
objMailAddress = CREATEOBJECT("AddEmail.MailAddress")
objMailAddress.Name = "John Smith"
objMailAddress.Address = "jsmith@myserver.com"
objMailMessage.Sender = objMailAddress

* Setup first recipient
objMailAddress = CREATEOBJECT("AddEmail.MailAddress")
objMailAddress.Name = "Jane Smith"
objMailAddress.Address = "jane@someserver.com"
objMailMessage.AddRecipient(objMailAddress)

* Setup second recipient
objMailAddress = CREATEOBJECT("AddEmail.MailAddress")
objMailAddress.Name = "James Smith"
objMailAddress.Address = "james@someserver.com"
objMailMessage.AddRecipient(objMailAddress)

* Send prepared message
numResultCode = objSmtpMail.Send(objMailMessage, .t., @strError)
IF numResultCode = 0 THEN
    MESSAGEBOX("Sent successfully!")
ELSE
    MESSAGEBOX(strError)
ENDIF
```

Please refer to the Reference section of this manual for detailed description of AddEmail objects, methods and properties.

2.14 PowerBuilder

2.14.1 Using AddEmail in PowerBuilder projects

AddEmail ActiveX can be used to send emails from PowerBuilder applications. To use AddEmail in your PowerBuilder project you need to add a code that creates SmtpMail object and sends a message using SimpleSend, SimpleSendAttachment or Send. Please use code snapshots below to get started.

Snapshot 1: simple text email.

```
OLEObject objSmtpMail
string strError
long numResultCode

objSmtpMail = Create OLEObject
If objSmtpMail.ConnectToNewObject("AddEmail.SmtpMail") < 0 Then
        Destroy objSmtpMail
        MessageBox("AddEmail", "Can't create AddEmail.SmtpMail COM object")
        Return
End If

objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpUsername = "jsmith"
objSmtpMail.SmtpPassword = "mypassword"
numResultCode = objSmtpMail.SimpleSend("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", Ref strError)
If numResultCode = 0 Then
    MessageBox("AddEmail", "Sent successfully!")
Else
    MessageBox("AddEmail", strError)
End If
Destroy objSmtpMail
```

Snapshot 2: text email with attachments.

```
OLEObject objSmtpMail
string strError
long numResultCode

objSmtpMail = Create OLEObject
If objSmtpMail.ConnectToNewObject("AddEmail.SmtpMail") < 0 Then
        Destroy objSmtpMail
        MessageBox("AddEmail", "Can't create AddEmail.SmtpMail COM object")
        Return
End If

objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpUsername = "jsmith"
objSmtpMail.SmtpPassword = "mypassword"
numResultCode = objSmtpMail.SimpleSendAttachment("jsmith@myserver.com",
"jane@someserver.com;james@someserver.com", "test", "Test message", "c:\files
\doc1.pdf;c:\files\doc2.pdf", Ref strError)
If numResultCode = 0 Then
    MessageBox("AddEmail", "Sent successfully!")
Else
    MessageBox("AddEmail", strError)
End If
Destroy objSmtpMail
```

Snapshot 2: HTML message with attachments.

```
OLEObject objSmtpMail, objMailMessage, objMailAttachment, objMailAddress
string strError
```

```
long numResultCode

objSmtpMail = Create OLEObject
If objSmtpMail.ConnectToNewObject("AddEmail.SmtpMail") < 0 Then
        Destroy objSmtpMail
        MessageBox("AddEmail", "Can't create AddEmail.SmtpMail COM object")
        Return
End If
objSmtpMail.SmtpServer = "mail.myserver.com"
objSmtpMail.SmtpUsername = "jsmith"
objSmtpMail.SmtpPassword = "mypassword"

objMailMessage = Create OLEObject
objMailMessage.ConnectToNewObject("AddEmail.MailMessage")
objMailMessage.MessageBodyFormat = 1
objMailMessage.MessageSubject = "test"
objMailMessage.MessageBody = "<html><body><b>Testing...</b></body></html>"

objMailAttachment = Create OLEObject
objMailAttachment.ConnectToNewObject("AddEmail.MailAttachment")
objMailAttachment.File = "c:\files\doc1.pdf"
objMailMessage.AddAttachment(objMailAttachment)

objMailAttachment = Create OLEObject
objMailAttachment.ConnectToNewObject("AddEmail.MailAttachment")
objMailAttachment.File = "c:\files\doc2.pdf"
objMailMessage.AddAttachment(objMailAttachment)

objMailAddress = Create OLEObject
objMailAddress.ConnectToNewObject("AddEmail.MailAddress")
objMailAddress.Name = "John Smith"
objMailAddress.Address = "jsmith@myserver.com"
objMailMessage.Sender = objMailAddress

objMailAddress.ConnectToNewObject("AddEmail.MailAddress")
objMailAddress.Name = "Jane Smith"
objMailAddress.Address = "jane@someserver.com"
objMailMessage.AddRecipient(objMailAddress)

objMailAddress.ConnectToNewObject("AddEmail.MailAddress")
objMailAddress.Name = "James Smith"
objMailAddress.Address = "james@someserver.com"
objMailMessage.AddRecipient(objMailAddress)

numResultCode = objSmtpMail.Send(objMailMessage, True, Ref strError)
If numResultCode = 0 Then
    MessageBox("AddEmail", "Sent successfully!")
Else
    MessageBox("AddEmail", strError)
End If
Destroy objSmtpMail
Destroy objMailMessage
Destroy objMailAddress
Destroy objMailAttachment
```

Please refer to the Reference section of this manual for detailed description of AddEmail objects, methods and properties.

III Reference

3.1 SmtpMail

3.1.1 Overview

SmtpMail object is used to send messages, obtain status of messages, cancel delivery of messages and set SMTP server parameters. **SmtpMail** object should remain in memory until all messages are either delivered, canceled or failed. An application can create more than one instance of **SmtpMail** object if it needs to.

Syntax:

[Visual Basic]

```
Dim objSmtpMail As New AddEmailLib.SmtpMail
objSmtpMail.SmtpServer = "mail.myisp.com"
```

[VBScript]

```
Dim objSmtpMail
Set objSmtpMail = CreateObject("AddEmail.SmtpMail")
objSmtpMail.SmtpServer = "mail.myisp.com"
```

[C#]

```
AddEmailLib.SmtpMailClass objSmtpMail = new AddEmailLib.SmtpMailClass();
objSmtpMail.SmtpServer = "mail.myisp.com";
```

[C++]

```
AddEmailLib::ISmtpMailPtr spSmtpMail;
spSmtpMail.CreateInstance(__uuidof(AddEmailLib::SmtpMail));
spSmtpMail->PutSmtpServer("mail.myisp.com");
```

Properties:

SmtpServer	Name or IP address of the SMTP server to use for sending e-mail messages.
SmtpPort	Port number on the SMTP server.
SmtpUsername	User name (account name) on the SMTP server.
SmtpPassword	Password on the SMTP server.
SmtpSSL	Specifies whether SMTP server requires an encrypted connection (TLS/ SSL).
SmtpSPA	Specifies whether to use Secure Password Authentication (SPA).
MaxThreads	Maximum number of mail sending threads.

SerialNumber	Serial number to activate purchased copy of AddEmail.
SenderHostname	Fully-qualified hostname of the computer that sends the e-mails.
ConnectAttempts	Number of connect attempts before reporting connection error.
ReplyTimeout	Number of seconds to wait for a response from SMTP server before reporting timeout error.
OnStatusChangeHandler	Handler for the OnStatusChange event.
OnProgressHandler	Handler for the OnProgress event.

Methods:

SimpleSend	Sends simple text or HTML e-mail message synchronously.
SimpleSendAttachment	Sends simple text or HTML e-mail message with attachments synchronously.
SimpleSendHtml	Imports HTML file and sends HTML e-mail with embedded images synchronously.
Send	Sends e-mail message synchronously.
SendAsync	Sends e-mail message asynchronously.
GetStatus	Returns status of e-mail message.
GetErrorCode	Returns error code of failed e-mail message.
GetErrorDescription	Returns error description of failed e-mail message.
Cancel	Cancels sending of queued e-mail message.
CancelAll	Cancels all queued e-mail messages.
GetLastErrorCode	Returns error code of the last synchronous send operation.
GetLastErrorDescription	Returns error description of the last synchronous send operation.
GetQueueSize	Returns number of e-mail messages queued for sending.

Events:

OnStatusChange	Notifies of message status changes.
OnProgress	Notifies of message sending progress.

3.1.2 SmtpServer Property

SmtpServer property specifies name or IP address of the outgoing mail server (SMTP server) to use for sending e-mail messages.

Syntax

[Visual Basic]

```
objSmtpMail.SmtpServer = "mail.myisp.com"
```

[VBScript]

```
objSmtpMail.SmtpServer = "mail.myisp.com"
```

[C#]

```
objSmtpMail.SmtpServer = "mail.myisp.com";
```

[C++]

```
HRESULT get_SmtpServer(BSTR*);
HRESULT put_SmtpServer(BSTR);

spSmtpMail->PutSmtpServer("mail.myisp.com");
```

Remarks

SmtpServer property can be empty if messages are to be sent directly, without using outgoing mail server (AddEmail Enterprise version only). Please refer to Send Method, SendAsync Method and **EnterpriseMail** sample for details.

3.1.3 SmtpPort Property

SmtpPort property specifies port number on the outgoing mail server.

Syntax

[Visual Basic]

```
objSmtpMail.SmtpPort = 25
```

[VBScript]

```
objSmtpMail.SmtpPort = 25
```

[C#]

```
objSmtpMail.SmtpPort = 25;
```

[C++]

```
HRESULT get_SmtpPort(LONG*);
HRESULT put_SmtpPort(LONG);

spSmtpMail->PutSmtpPort(25);
```

Remarks

Default value of **SmtpPort** property is 25 and doesn't have to be changed for most SMTP servers. If port 25 doesn't work, please ask your network administrator or internet service provider what port should be used to send e-mails.

3.1.4 SmtpUsername Property

SmtpUsername property specifies username (login or account name) on the outgoing mail server.

Syntax

[Visual Basic]

```
objSmtpMail.SmtpUsername = "smithj"
```

[VBScript]

```
objSmtpMail.SmtpUsername = "smithj"
```

[C#]

```
objSmtpMail.SmtpUsername = "smithj";
```

[C++]

```
HRESULT get_SmtpUsername(BSTR*);
HRESULT put_SmtpUsername(BSTR);

spSmtpMail->PutSmtpUsername("smithj");
```

Remarks

SmtpUsername property contains empty string by default. This property should be set to an empty string if your outgoing mail server doesn't require authentication.

3.1.5 SmtpPassword Property

SmtpPassword property specifies password on the outgoing mail server.

Syntax

[Visual Basic]

```
objSmtpMail.SmtpPassword = "pass"
```

[VBScript]

```
objSmtpMail.SmtpPassword = "pass"
```

[C#]

```
objSmtpMail.SmtpPassword = "pass";
```

[C++]

```
HRESULT get_SmtpPassword(BSTR*);
HRESULT put_SmtpPassword(BSTR);

spSmtpMail->PutSmtpPassword("pass");
```

Remarks

SmtpPassword property contains empty string by default. This property should be set to an empty string if your outgoing mail server doesn't require authentication.

3.1.6 SmtpSSL Property

SmtpSSL property specifies whether outgoing mail server requires an encrypted connection (TLS/SSL).

Syntax

[Visual Basic]

```
objSmtpMail.SmtpSSL = True
```

[VBScript]

```
objSmtpMail.SmtpSSL = True
```

[C#]

```
objSmtpMail.SmtpSSL = true;
```

[C++]

```
HRESULT get_SmtpSSL(VARIANT_BOOL*);
HRESULT put_SmtpSSL(VARIANT_BOOL);

spSmtpMail->PutSmtpSSL(VARIANT_TRUE);
```

Remarks

SmtpSSL property is *False* by default. Set **SmtpSSL** property to *True* if you want to use TLS/SSL connection, for example when using *smtp.gmail.com* server on port 465.

3.1.7 SmtpSPA Property

SmtpSPA property specifies whether to use Secure Password Authentication (SPA) method. SPA is supported by Microsoft Exchange, Windows 2000 Server and Windows 2003 Server.

Syntax

[Visual Basic]

```
objSmtpMail.SmtpSPA = True
```

[VBScript]

```
objSmtpMail.SmtpSPA = True
```

[C#]

```
objSmtpMail.SmtpSPA = true;
```

[C++]

```
HRESULT get_SmtpSPA(VARIANT_BOOL*);
HRESULT put_SmtpSPA(VARIANT_BOOL);

spSmtpMail->PutSmtpSPA(VARIANT_TRUE);
```

Remarks

SmtpSPA property is *False* by default. If your outgoing mail server requires SPA please set **SmtpSPA** property to *True* in order to use Secure Password Authentication. Please note that some servers are configured to require SmtpUsername in *username@domain.com* format for SPA authentication.

3.1.8 MaxThreads Property

MaxThreads property specifies the maximum number of threads that **SmtpMail** object can create when sending messages asynchronously using SendAsync (Enterprise version only).

Syntax

[Visual Basic]

```
objSmtpMail.MaxThreads = 10
```

[VBScript]

```
objSmtpMail.MaxThreads = 10
```

[C#]

```
objSmtpMail.MaxThreads = 10;
```

[C++]

```
HRESULT get_MaxThreads(LONG*);
HRESULT put_MaxThreads(LONG);

spSmtpMail->PutMaxThreads(10);
```

Remarks

Default value of **MaxThreads** property is 5, accepted range is 1 - 1000. AddEmail supports simultaneous sending of several messages. This feature is useful if application has to send a large number of messages, especially if messages are sent directly without using outgoing mail server. **SmtpMail** object creates one or more worker threads that send messages from its message queue. **MaxThreads** property limits the number of threads that are running at the same time, therefore setting the limit on the number of simultaneously transferred messages. **MaxThreads** property is available only in AddEmail Enterprise version. AddEmail Professional always uses one thread to send messages.

3.1.9 SenderHostname Property

Use **SenderHostname** property to specify fully-qualified hostname of the computer that sends e-mail messages (Enterprise version only). **SenderHostname** is used when AddEmail sends e-mails directly without using SMTP server of your organization or internet provider. Please see remarks for more information.

Syntax

[Visual Basic]

```
objSmtpMail.SenderHostname = "myserver.mydomain.com"
```

[VBScript]

```
objSmtpMail.SenderHostname = "myserver.mydomain.com"
```

[C#]

```
objSmtpMail.SenderHostname = "myserver.mydomain.com";
```

[C++]

```
HRESULT get_SenderHostname(BSTR*);
HRESULT put_SenderHostname(BSTR);

spSmtpMail->PutSenderHostname("myserver.mydomain.com");
```

Remarks

SenderHostname property should be set to the fully-qualified hostname of the computer that sends e-mail messages, for example *myserver.mydomain.com*. **SenderHostname** is transmitted to the recipient's mail server when AddEmail sends e-mail messages directly. Some mail servers do not accept incoming mail if **SenderHostname** is not set. In addition, some mail servers check reverse DNS entry for the IP address of the computer that sends e-mail messages directly and verify that it matches with the **SenderHostname**. Use the following link to check reverse DNS entry for an IP address: http://postmaster.aol.com/cgi-bin/plugh/rdns.pl

3.1.10 SimpleSend Method

SimpleSend method sends simple text or HTML e-mail to one or more recipients synchronously.

Syntax

[Visual Basic]

```
Function SimpleSend(strFrom As String, strTo As String, strSubject As String,
strMessage As String, ByRef strError As String) As Long
```

Example:

```
Dim strError As String
```

```
Dim numResultCode As Long
numResultCode = objSmtpMail.SimpleSend("My Name <me@myisp.com>", "Someone
<someone@someisp.com>;someoneelse@someotherisp.com", "test", "Test message", strError)
```

[VBScript]

```
Function SimpleSendScriptable(strFrom, strTo, strSubject, strMessage, ByRef strError)
```

Example:

```
Dim strError
Dim numResultCode
numResultCode = objSmtpMail.SimpleSendScriptable("My Name <me@myisp.com>", "Someone
<someone@someisp.com>;someoneelse@someotherisp.com", "test", "Test message", strError)
```

[C#]

```
int SimpleSend(string strFrom, string strTo, string strSubject, string strMessage, out
string strError);
```

Example:

```
string strError;
int numResultCode = objSmtpMail.SimpleSend("My Name <me@myisp.com>", "Someone
<someone@someisp.com>;someoneelse@someotherisp.com", "test", "Test message", out
strError);
```

[C++]

```
HRESULT SimpleSend(BSTR strFrom, BSTR strTo, BSTR strSubject, BSTR strMessage, BSTR*
pstrError, LONG* pnumResultCode);
```

Example:

```
BSTR bstr;
int numResultCode = spSmtpMail->SimpleSend("My Name <me@myisp.com>", "Someone
<someone@someisp.com>;someoneelse@someotherisp.com", "test", "Test message", &bstr);
_bstr_t strError(bstr, FALSE);
```

Parameters

`strFrom` *[in]*
Sender's e-mail address. You can also specify sender's name followed by the sender's e-mail address in angled brackets.

`strTo` *[in]*
List of recipients' e-mail addresses separated by semicolon. You can also specify recipient's name followed by the recipient's e-mail address in angled brackets.

`strSubject` *[in]*
Subject of the e-mail message.

`strMessage` *[in]*
Body of the e-mail message.

`strError` *[out]*
Contains extended error information if the message failed to be sent, empty string otherwise.

Return value

Returns numeric result:
0 if the message was sent successfully;
-1 if connection to the server failed or in case of other Winsock errors;
>0 SMTP error code as defined in RFC 821 page 34-35.

Remarks

SimpleSend can be used to send text or HTML e-mails. Body of the e-mail message should start with <html> tag for HTML e-mail messages. This method supports Unicode in the subject and body of the message as well as in the sender's and recipients' names. AddEmail will create Unicode e-mail (UTF-8 encoding) if any part of the message contains Unicode characters.

This method is synchronous: it returns after the message was sent or after an error occuried. Please use it with care because single-threaded applications are blocked and can't process user input for the duration of the **SimpleSend** call. Please refer to SendAsync Method if you want to send messages without blocking your application.

3.1.11 SimpleSendAttachment Method

SimpleSendAttachment method sends simple text or HTML e-mail with attachments to one or more recipients synchronously.

Syntax

[Visual Basic]

```
Function SimpleSendAttachment(strFrom As String, strTo As String, strSubject As
String, strMessage As String, strAttach As String, ByRef strError As String) As Long
```

Example:

```
Dim strError As String
Dim numResultCode As Long
numResultCode = objSmtpMail.SimpleSendAttachment("My Name <me@myisp.com>", "Someone
<someone@someisp.com>;someoneelse@someotherisp.com", "test", "Test message", "c:\files
\doc1.pdf;c:\files\doc2.pdf", strError)
```

[VBScript]

```
Function SimpleSendAttachmentScriptable(strFrom, strTo, strSubject, strMessage,
strAttach, ByRef strError)
```

Example:

```
Dim strError
Dim numResultCode
numResultCode = objSmtpMail.SimpleSendAttachmentScriptable("My Name <me@myisp.com>",
```

```
"Someone <someone@someisp.com>;someoneelse@someotherisp.com", "test", "Test message",
"c:\files\doc1.pdf;c:\files\doc2.pdf", strError)
```

[C#]

```
int SimpleSendAttachment(string strFrom, string strTo, string strSubject, string
strMessage, string strAttach, out string strError);
```

Example:

```
string strError;
int numResultCode = objSmtpMail.SimpleSendAttachment("My Name <me@myisp.com>",
"Someone <someone@someisp.com>;someoneelse@someotherisp.com", "test", "Test message",
"c:\\files\\doc1.pdf;c:\\files\\doc2.pdf", out strError);
```

[C++]

```
HRESULT SimpleSendAttachment(BSTR strFrom, BSTR strTo, BSTR strSubject, BSTR
strMessage, BSTR strAttach, BSTR* pstrError, LONG* pnumResultCode);
```

Example:

```
BSTR bstr;
int numResultCode = spSmtpMail->SimpleSendAttachment("My Name <me@myisp.com>",
"Someone <someone@someisp.com>;someoneelse@someotherisp.com", "test", "Test message",
"c:\\files\\doc1.pdf;c:\\files\\doc2.pdf", &bstr);
_bstr_t strError(bstr, FALSE);
```

Parameters

`strFrom` *[in]*
Sender's e-mail address. You can also specify sender's name followed by the sender's e-mail address in angled brackets.

`strTo` *[in]*
List of recipients' e-mail addresses separated by semicolon. You can also specify recipient's name followed by the recipient's e-mail address in angled brackets.

`strSubject` *[in]*
Subject of the e-mail message.

`strMessage` *[in]*
Body of the e-mail message.

`strAttach` *[in]*
List of attachments separated by semicolon.

`strError` *[out]*
Contains extended error information if the message failed to be sent, empty string otherwise.

Return value

Returns numeric result:
0 if the message was sent successfully;

-1 if connection to the server failed or in case of other Winsock errors;
>0 SMTP error code as defined in RFC 821 page 34-35.

Remarks

SimpleSendAttachment can be used to send text or HTML e-mails. Body of the e-mail message should start with <html> tag for HTML e-mail messages. This method supports Unicode in the subject and body of the message as well as in the sender's and recipients' names. AddEmail will create Unicode e-mail (UTF-8 encoding) if any part of the message contains Unicode characters.

This method is synchronous: it returns after the message was sent or after an error occuried. Please use it with care because single-threaded applications are blocked and can't process user input for the duration of the **SimpleSendAttachment** call. Please refer to SendAsync Method if you want to send messages without blocking your application.

3.1.12 SimpleSendHtml Method

SimpleSendHtml method imports HTML file from disk and sends HTML e-mail with embedded images to one or more recipients synchronously.

Syntax

[Visual Basic]

```
Function SimpleSendHtml(strFrom As String, strTo As String, strSubject As String,
strHtmlFile As String, ByRef strError As String) As Long
```

Example:

```
Dim strError As String
Dim numResultCode As Long
numResultCode = objSmtpMail.SimpleSendHtml("My Name <me@myisp.com>", "Someone
<someone@someisp.com>;someoneelse@someotherisp.com", "test", "c:\files\email.html",
strError)
```

[VBScript]

```
Function SimpleSendHtmlScriptable(strFrom, strTo, strSubject, strHtmlFile, ByRef
strError)
```

Example:

```
Dim strError
Dim numResultCode
numResultCode = objSmtpMail.SimpleSendHtmlScriptable("My Name <me@myisp.com>",
"Someone <someone@someisp.com>;someoneelse@someotherisp.com", "test", "c:\files
\email.html", strError)
```

[C#]

```
int SimpleSendHtml(string strFrom, string strTo, string strSubject, string
strHtmlFile, out string strError);
```

Example:

```
string strError;
int numResultCode = objSmtpMail.SimpleSendHtml("My Name <me@myisp.com>", "Someone
<someone@someisp.com>;someoneelse@someotherisp.com", "test", "c:\\files\\email.html",
out strError);
```

[C++]

```
HRESULT SimpleSendHtml(BSTR strFrom, BSTR strTo, BSTR strSubject, BSTR strHtmlFile,
BSTR* pstrError, LONG* pnumResultCode);
```

Example:

```
BSTR bstr;
int numResultCode = spSmtpMail->SimpleSendHtml("My Name <me@myisp.com>", "Someone
<someone@someisp.com>;someoneelse@someotherisp.com", "test", "c:\\files\\email.html",
&bstr);
_bstr_t strError(bstr, FALSE);
```

Parameters

`strFrom` *[in]*
Sender's e-mail address. You can also specify sender's name followed by the sender's e-mail address in angled brackets.

`strTo` *[in]*
List of recipients' e-mail addresses separated by semicolon. You can also specify recipient's name followed by the recipient's e-mail address in angled brackets.

`strSubject` *[in]*
Subject of the e-mail message.

`strFileName` *[in]*
Filename of the HTML file to be imported.

`strError` *[out]*
Contains extended error information if the message failed to be sent, empty string otherwise.

Return value

Returns numeric result:
0 if the message was sent successfully;
-1 if connection to the server failed or in case of other Winsock errors;
>0 SMTP error code as defined in RFC 821 page 34-35.

Remarks

SimpleSendHtml can be used to send HTML e-mails with embedded images. This method imports specified HTML file, adds all images referenced in the HTML as inline attachments and modifies HTML as needed. Only images located on disk are added as attachments. Images from the web (http:// or https:// links in HTML) are not attached. Please note that the current version of AddEmail does not import images specified in CSS using url values.

This method supports Unicode in the subject and body of the message as well as in the sender's and recipients' names. Please make sure HTML file uses UTF-8 encoding if you want to use Unicode. AddEmail will create Unicode e-mail (UTF-8 encoding) if any part of the message contains Unicode characters.

This method is synchronous: it returns after the message was sent or after an error occuried. Please use it with care because single-threaded applications are blocked and can't process user input for the duration of the **SimpleSend** call. Please refer to SendAsync Method if you want to send messages without blocking your application.

3.1.13 SerialNumber Property

SerialNumber property specifies serial number to activate purchased copy of AddEmail.

Syntax

[Visual Basic]

```
objSmtpMail.SerialNumber = "12345-67890-ABCDE-EFGH"
```

[VBScript]

```
objSmtpMail.SerialNumber = "12345-67890-ABCDE-EFGH"
```

[C#]

```
objSmtpMail.SerialNumber = "12345-67890-ABCDE-EFGH";
```

[C++]

```
HRESULT get_SerialNumber(BSTR*);
HRESULT put_SerialNumber(BSTR);

spSmtpMail->PutSerialNumber("12345-67890-ABCDE-EFGH");
```

Remarks

SerialNumber property contains an empty string by default. After purchasing AddEmail you will receive serial number needed to activate AddEmail. Set **SerialNumber** property to the serial number you received.

3.1.14 ConnectAttempts Property

ConnectAttempts specifies how many times AddEmail will try to connect to the outgoing mail server before returning connection error.

Syntax

[Visual Basic]

```
objSmtpMail.ConnectAttempts = 1
```

[VBScript]

```
objSmtpMail.ConnectAttempts = 1
```

[C#]

```
objSmtpMail.ConnectAttempts = 1;
```

[C++]

```
HRESULT get_ConnectAttempts(LONG*);
HRESULT put_ConnectAttempts(LONG);

spSmtpMail->PutConnectAttempts(1);
```

Remarks

Default value of **ConnectAttempts** property is 2. To send an e-mail, AddEmail connects to the outgoing mail server specified in SmtpServer property. If connection could not be established, AddEmail will make **ConnectAttempts** attempts to connect before giving up and reporting connection error. If the server is down and does not respond, each connect attempt takes 20-30 seconds. You might want to set **ConnectAttempts** to 1 if you are sending e-mails directly without using SMTP server in order to speed up sending.

3.1.15 ReplyTimeout Property

ReplyTimeout specifies how many seconds AddEmail will wait for a reply from the outgoing mail server before returning timeout error.

Syntax

[Visual Basic]

```
objSmtpMail.ReplyTimeout = 20
```

[VBScript]

```
objSmtpMail.ReplyTimeout = 20
```

[C#]

```
objSmtpMail.ReplyTimeout = 20;
```

[C++]

```
HRESULT get_ReplyTimeout(LONG*);
HRESULT put_ReplyTimeout(LONG);

spSmtpMail->PutReplyTimeout(20);
```

Remarks

Default value of **ReplyTimeout** property is 30. To send an e-mail, AddEmail sends SMTP commands to the outgoing mail server specified in SmtpServer property. After each sent command AddEmail will wait up to **ReplyTimeout** seconds for a reply from the server. If the server does not respond within specified **ReplyTimeout**, connection to the server is terminated and timeout error is reported. You might want to set **ReplyTimeout** to 60-120 seconds if your outgoing mail server is slow and you get timeout errors with the default value.

3.1.16 Send Method

Send method sends e-mail message synchronously.

Syntax

[Visual Basic]

```
Function Send(objMessage As AddEmailLib.MailMessage, bUseSmtpServer As Boolean, ByRef
strError As String) As Long
```

Example:

```
Dim strError As String
Dim numResultCode As Long
numResultCode = objSmtpMail.Send(objMessage, True, strError)
```

[VBScript]

```
Function SendScriptable(objMessage, bUseSmtpServer, ByRef strError)
```

Example:

```
Dim strError
Dim numResultCode
numResultCode = objSmtpMail.SendScriptable(objMessage, True, strError)
```

[C#]

```
int Send(AddEmailLib.MailMessageClass objMessage, bool bUseSmtpServer, out string
strError);
```

Example:

```
string strError;
int numResultCode = objSmtpMail.Send(objMessage, true, out strError);
```

[C++]

```
HRESULT Send(IMailMessage* objMessage, VARIANT_BOOL bUseSmtpServer, BSTR* pstrError,
LONG* pnumResultCode);
```

Example:

```
BSTR bstr;
int numResultCode = spSmtpMail->Send(objMessage, TRUE, &bstr);
_bstr_t strError(bstr, FALSE);
```

Parameters

`objMessage` *[in]*
Prepared MailMessage object containing the e-mail message to send.

`bUseSmtpServer` *[in]*
Specifies whether the message should be sent using SMTP server. If the parameter is true (not equal to zero) message will be send using SMTP server specified in SmtpServer property. If the parameter is false (equal to zero) message will be send directly to the recipients' mail server(s). This parameter is valid for AddEmail Enterprise version only. AddEmail Professional always uses specified SMTP server.

`strError` *[out]*
Contains extended error information if the message failed to be sent, empty string otherwise.

Return value

Returns numeric result:
0 if message was sent successfully;
-1 if connection to the server failed or in case of other Winsock errors;
>0 SMTP error code as defined in RFC 821 page 34-35.

Remarks

This method is synchronous: it returns after the message was sent or after an error occuried. Please use it with care because single-threaded applications are blocked and can't process user input for the duration of **Send** call. Please refer to SendAsync Method if you want to send messages without blocking your application. Please refer to the SenderHostname topic for more information about sending e-mails directly without using SMTP server of your organization or internet provider (Enterprise version only).

3.1.17 SendAsync Method

SendAsync method sends e-mail message asynchronously.

Syntax

[Visual Basic]

```
Function SendAsync(objMessage As AddEmailLib.MailMessage, bUseSmtpServer As Boolean)
As Long
```

Example:

```
Dim numMessageNumber As Long
numMessageNumber = objSmtpMail.SendAsync(objMessage, True)
```

[VBScript]

```
Function SendAsync(objMessage, bUseSmtpServer)
```

Example:

```
Dim numMessageNumber
numMessageNumber = objSmtpMail.SendAsync(objMessage, True)
```

[C#]

```
int SendAsync(AddEmailLib.MailMessageClass objMessage, bool bUseSmtpServer);
```

Example:

```
int numMessageNumber = objSmtpMail.SendAsync(objMessage, true);
```

[C++]

```
HRESULT SendAsync(IMailMessage* objMessage, VARIANT_BOOL bUseSmtpServer, LONG*
pnumResultCode);
```

Example:

```
int numMessageNumber = spSmtpMail->SendAsync(objMessage, TRUE);
```

Parameters

`objMessage` *[in]*
Prepared MailMessage object containing the e-mail message to send.

`bUseSmtpServer` *[in]*
Specifies whether the message should be sent using SMTP server. If the parameter is true (not equal to zero), message will be send using SMTP server specified in SmtpServer property. If the parameter is false (equal to zero), message will be send directly to the recipients' mail server(s). This parameter is valid for AddEmail Enterprise version only. AddEmail Professional always uses specified SMTP server.

Return value

Returns message number which identifies this message. Message number is used to obtain status of the message or cancel the message.

Remarks

This method is asynchronous: it validates the message, places it to the message queue and returns immediately. Actual sending of the message is done in another thread. **SendAsync** doesn't block an application and can be safely used in single-threaded applications. The message number returned by this method is used in GetStatus Method, GetErrorCode Method, GetErrorDescription Method, Cancel Method, OnStatusChange Event and OnProgress Event. Please refer to the SenderHostname topic for more information about sending e-mails directly without using SMTP server of your organization or internet provider (Enterprise version only).

3.1.18 GetStatus Method

GetStatus method obtains status of a message that was queued for sending using SendAsync method.

Syntax

[Visual Basic]

```
Function GetStatus(numMessageNumber As Long) As AddEmailLib.MailStatus
```

Example:

```
Dim messageStatus As AddEmailLib.MailStatus
messageStatus = objSmtpMail.GetStatus(numMessageNumber)
```

[VBScript]

```
Function GetStatus(numMessageNumber)
```

Example:

```
Dim messageStatus
messageStatus = objSmtpMail.GetStatus(numMessageNumber)
```

[C#]

```
AddEmailLib.MailStatus GetStatus(int numMessageNumber);
```

Example:

```
AddEmailLib.MailStatus messageStatus = objSmtpMail.GetStatus(numMessageNumber);
```

[C++]

```
HRESULT GetStatus(LONG numMessageNumber, MailStatus* pmessageStatus);
```

Example:

```
AddEmailLib::MailStatus messageStatus = spSmtpMail->GetStatus(numMessageNumber);
```

Parameters

`numMessageNumber` *[in]*
Message number returned from the SendAsync method.

Return value

Returns status of the e-mail message. Message status values are defined in MailStatus enumeration.

Remarks

SendAsync method places a message into a message queue and returns message number which identifies the message. Use **GetStatus** method to obtain information about a status of the message.

3.1.19 GetErrorCode Method

GetErrorCode method obtains error code for a message that was queued for sending using SendAsync method and failed to be sent.

Syntax

[Visual Basic]

```
Function GetErrorCode(numMessageNumber As Long) As Long
```

Example:

```
numErrorCode = objSmtpMail.GetErrorCode(numMessageNumber)
```

[VBScript]

```
Function GetErrorCode(numMessageNumber)
```

Example:

```
numErrorCode = objSmtpMail.GetErrorCode(numMessageNumber)
```

[C#]

```
int GetErrorCode(int numMessageNumber);
```

Example:

```
int numErrorCode = objSmtpMail.GetErrorCode(numMessageNumber);
```

[C++]

```
HRESULT GetErrorCode(LONG numMessageNumber, LONG* pnumErrorCode);
```

Example:

```
int numErrorCode = spSmtpMail->GetErrorCode(numMessageNumber);
```

Parameters

`numMessageNumber` *`[in]`*
Message number returned from the SendAsync method.

Return value

Returns numeric error code:
0 if message was sent successfully;
-1 if connection to the server failed or in case of other Winsock errors;
>0 SMTP error code as defined in RFC 821 page 34-35.

Remarks

SendAsync method places a message into a message queue and returns message number which identifies the message. If the message failed to be sent status of the message becomes **MailStatusFailed** and **GetErrorCode** method can be used to obtain error code.

3.1.20 GetErrorDescription Method

GetErrorDescription method obtains error description for a message that was queued for sending using SendAsync method and failed to be sent.

Syntax

[Visual Basic]

```
Function GetErrorDescription(numMessageNumber As Long) As String
```

Example:

```
Dim strError As String
strError = objSmtpMail.GetErrorDescription(numMessageNumber)
```

[VBScript]

```
Function GetErrorDescription(numMessageNumber)
```

Example:

```
Dim strError
strError = objSmtpMail.GetErrorDescription(numMessageNumber)
```

[C#]

```
string GetErrorDescription(int numMessageNumber);
```

Example:

```
string strError = objSmtpMail.GetErrorDescription(numMessageNumber);
```

[C++]

```
HRESULT GetErrorDescription(LONG numMessageNumber, BSTR* pstrError);
```

Example:

```
_bstr_t strError = spSmtpMail->GetErrorDescription(numMessageNumber);
```

Parameters

numMessageNumber *[in]*
Message number returned from the SendAsync method.

Return value

Returns extended error information if the message failed to be sent, empty string otherwise.

Remarks

SendAsync method places a message into a message queue and returns message number which identifies the message. If message failed to be sent, status of the message becomes **MailStatusFailed** and **GetErrorDescription** method can be used to obtain error description.

3.1.21 Cancel Method

Cancel method cancels sending of a message that was queued for sending using SendAsync method.

Syntax

[Visual Basic]

```
Sub Cancel(numMessageNumber As Long)
```

Example:

```
objSmtpMail.Cancel numMessageNumber
```

[VBScript]

```
Sub Cancel(numMessageNumber)
```

Example:

```
objSmtpMail.Cancel numMessageNumber
```

[C#]

```
void Cancel(int numMessageNumber);
```

Example:

```
objSmtpMail.Cancel(numMessageNumber);
```

[C++]

```
HRESULT Cancel(LONG numMessageNumber);
```

Example:

```
spSmtpMail->Cancel(numMessageNumber);
```

Parameters

`numMessageNumber` *[in]*
Message number returned from the SendAsync method.

Return value

None.

Remarks

SendAsync method places a message into a message queue and returns message number which identifies the message. Use **Cancel** method to cancel sending of the message. Only messages that have status **MailStatusQueued** or **MailStatusSending** can be canceled. If the message is currently transferred and its status is **MailStatusSending**, **Cancel** method doesn't cause the transfer to stop immediately. Message is marked for cancellation and will be canceled some time later. The message is actually canceled when its status changes to **MailStatusCanceled**.

3.1.22 CancelAll Method

CancelAll method cancels sending of all messages that were queued for sending using SendAsync method.

Syntax

[Visual Basic]

```
Sub CancelAll()
```

Example:

```
objSmtpMail.CancelAll
```

[VBScript]

```
Sub CancelAll()
```

Example:

```
objSmtpMail.CancelAll
```

[C#]

```
void Cancel();
```

Example:

```
objSmtpMail.CancelAll();
```

[C++]

```
HRESULT CancelAll();
```

Example:

```
spSmtpMail->CancelAll();
```

Parameters

None.

Return value

None.

Remarks

SendAsync method places a message into a message queue and returns message number which identifies the message. Use **CancelAll** method to cancel sending of all messages that were not sent yet. Only messages that have status **MailStatusQueued** or **MailStatusSending** will be canceled. If a message is currently transferred and its status is **MailStatusSending**, **CancelAll** method doesn't cause the transfer to stop immediately. Message is marked for cancellation and will be canceled some time later. The message is actually canceled when its status changes to **MailStatusCanceled**.

3.1.23 GetLastErrorCode Method

GetLastErrorCode method returns error code of the last synchronous send operation.

Syntax

[Visual Basic]

```
Function GetLastErrorCode() As Long
```

Example:

```
numErrorCode = objSmtpMail.GetLastErrorCode
```

[VBScript]

```
Function GetLastErrorCode()
```

Example:

```
numErrorCode = objSmtpMail.GetLastErrorCode
```

[C#]

```
int GetLastErrorCode();
```

Example:

```
int numErrorCode = objSmtpMail.GetLastErrorCode();
```

[C++]

```
HRESULT GetLastErrorCode(LONG* pnumErrorCode);
```

Example:

```
int numErrorCode = spSmtpMail->GetLastErrorCode();
```

Parameters

None.

Return value

Returns numeric error code:
0 if the last message was sent successfully;
-1 if connection to the server failed or in case of other Winsock errors;
>0 SMTP error code as defined in RFC 821 page 34-35.

Remarks

GetLastErrorCode can be used to obtain error code of the last synchronous send operation, i.e. the error code from the last call of Send, SimpleSend, SimpleSendAttachment, SimpleSendScriptable or SimpleSendAttachmentScriptable.

3.1.24 GetLastErrorDescription Method

GetLastErrorDescription method returns error description of the last synchronous send operation.

Syntax

[Visual Basic]

```
Function GetLastErrorDescription() As String
```

Example:

```
Dim strError As String
strError = objSmtpMail.GetLastErrorDescription
```

[VBScript]

```
Function GetLastErrorDescription()
```

Example:

```
Dim strError
strError = objSmtpMail.GetLastErrorDescription
```

[C#]

```
string GetLastErrorDescription();
```

Example:

```
string strError = objSmtpMail.GetLastErrorDescription();
```

[C++]

```
HRESULT GetLastErrorDescription(BSTR* pstrError);
```

Example:

```
_bstr_t strError = spSmtpMail->GetLastErrorDescription();
```

Parameters

None.

Return value

Returns extended error information if the last message failed to be sent, empty string otherwise.

Remarks

If programming language you are using does not support out parameters (JavaScript for example), **GetLastErrorDescription** can be used to obtain extended error information of the last synchronous send operation, i.e. the error description from the last call of Send, SimpleSend, SimpleSendAttachment, SimpleSendScriptable or SimpleSendAttachmentScriptable.

3.1.25 GetQueueSize Method

GetQueueSize method returns number of e-mail messages queued for sending.

Syntax

[Visual Basic]

```
Function GetQueueSize() As Long
```

Example:

```
numMessages = objSmtpMail.GetQueueSize
```

[VBScript]

```
Function GetQueueSize()
```

Example:

```
numMessages = objSmtpMail.GetQueueSize
```

[C#]

```
int GetQueueSize();
```

Example:

```
int numMessages = objSmtpMail.GetQueueSize();
```

[C++]

```
HRESULT GetQueueSize(LONG* pnumErrorCode);
```

Example:

```
int numMessages = spSmtpMail->GetQueueSize();
```

Parameters

None.

Return value

Returns number of messages in AddEmail queue.

Remarks

SendAsync method places a message into a message queue and exits. Afer that AddEmail starts sending messages from the queue using background thread(s). Use **GetQueueSize** method to get the number of messages queued for sending.

3.1.26 OnStatusChange Event

OnStatusChange event is fired on every status change for the message that was queued for sending using SendAsync method.

Syntax

[Visual Basic]

```
Sub OnStatusChange(numMessageNumber As Long, numNewStatus As Long)
```

[C#]

```
void OnStatusChange(int numMessageNumber, int numNewStatus);
```

[C++]

```
void __cdecl OnStatusChange(LONG numMessageNumber, LONG numNewStatus );
```

Parameters

`numMessageNumber` *[in]*
Message number returned from the SendAsync method.

`numNewStatus` *[in]*
New status of the message, one of the values from the MailStatus enumeration.

Return value

None.

Remarks

SendAsync method places a message into a message queue and returns message number which identifies the message. **OnStatusChange** event is used to notify application about changes to the message status. Initially message status is **MailStatusQueued**. When AddEmail starts sending the message its status changes to **MailStatusSending**. When the message is sent successfully status changes to **MailStatusSent**. Status becomes **MailStatusFailed** if AddEmail was unable to send the message. If the message was canceled using Cancel or CancelAll method its status becomes **MailStatusCanceled**.

3.1.27 OnProgress Event

OnProgress event is fired during message transfer to notify application about sending progress.

Syntax

[Visual Basic]

```
Sub OnProgress(numMessageNumber As Long, numBytesSent As Long, numBytesTotal As Long)
```

[C#]

```
void OnProgress(int numMessageNumber, int numBytesSent, int numBytesTotal);
```

[C++]

```
void __cdecl OnProgress(LONG numMessageNumber, LONG numBytesSent, LONG numBytesTotal);
```

Parameters

`numMessageNumber` *[in]*
Message number returned from the SendAsync method.

`numBytesSent` *[in]*
Number of bytes transferred already.

`numBytesTotal` *[in]*
Total number of bytes in the message.

Return value

None.

Remarks

OnProgress is fired after block of data is transferred. In current version of AddEmail block size is 8 kilobytes. If the total number of bytes in the message is smaller than the block size, **OnProgress** event is not fired.

3.1.28 OnStatusChangeHandler Property

OnStatusChangeHandler property specifies a handler for the OnStatusChange event.

Syntax

[JavaScript]

```
smtpMail.OnStatusChangeHandler = MailOnStatusChange;
...

function MailOnStatusChange(mailNumber, status)
{
  ...
}
```

Remarks

Use **OnStatusChangeHandler** property to set a handler for the OnStatusChange event in script languages such as VBScript or JavaScript.

3.1.29 OnProgressHandler Property

OnProgressHandler property specifies a handler for the OnProgress event.

Syntax

[JavaScript]

```
smtpMail.OnProgressHandler = MailOnProgress;
...

function MailOnProgress(mailNumber, bytesSent, bytesTotal)
{
  ...
}
```

Remarks

Use **OnProgressHandler** property to set a handler for the OnProgress event in script languages such as VBScript or JavaScript.

3.2 MailMessage

3.2.1 Overview

MailMessage object represents e-mail message and contains all message data: sender, recipients, subject, body, attachments etc.

Syntax:

[Visual Basic]

```
Dim objMailMessage As New AddEmailLib.MailMessage
objMailMessage.MessageSubject = "test"
```

[VBScript]

```
Dim objMailMessage
Set objMailMessage = CreateObject("AddEmail.MailMessage")
objMailMessage.MessageSubject = "test"
```

[C#]

```
AddEmailLib.MailMessageClass objMailMessage = new AddEmailLib.MailMessageClass();
objMailMessage.MessageSubject = "test";
```

[C++]

```
AddEmailLib::IMailMessagePtr spMailMessage;
spMailMessage.CreateInstance(__uuidof(AddEmailLib::MailMessage));
spMailMessage->PutMessageSubject("test");
```

Properties:

Sender	Sender of e-mail message.
ReplyTo	Reply-to address.
MessageSubject	Subject of e-mail message.
MessageBody	Body of e-mail message.
MessageBodyFormat	Format of the body of e-mail message.
MessageBodyEncoding	Encoding of the body of e-mail message.
MessageBodyCharset	Character set of the body of e-mail message.
AltMessageBody	Alternative text-only body of e-mail message.
AltMessageBodyEncoding	Encoding of the alternative body of e-mail message.
AltMessageBodyCharset	Character set of the alternative body of e-mail message.
MessagePriority	Priority of e-mail message.
MessageId	Id of e-mail message.
MessageBodyFile	Filename of the file that contains body of e-mail message.
RequestReadReceipt	Request a read receipt for the message.

Methods:

AddRecipient	Adds recipient to the To field of e-mail message.
AddCcRecipient	Adds recipient to the Cc field of e-mail message.
AddBccRecipient	Adds recipient to the Bcc field of e-mail message.
AddAttachment	Adds attachment to e-mail message.

AddHeader	Adds header value to e-mail message.
ClearAttachments	Clears list of attachments.
ClearRecipients	Clears list of recipients in the To field of e-mail message.
ClearCcRecipients	Clears list of recipients in the Cc field of e-mail message.
ClearBccRecipients	Clears list of recipients in the Bcc field of e-mail message.
ImportHTML	Imports HTML file with embedded images.

3.2.2 Sender Property

Sender property specifies sender of e-mail message.

Syntax

[Visual Basic]

```
Dim objMailAddress As New AddEmailLib.MailAddress
objMailAddress.Address = "me@myisp.com"
objMailMessage.Sender = objMailAddress
```

[VBScript]

```
Dim objMailAddress
Set objMailAddress = CreateObject("AddEmail.MailAddress")
objMailAddress.Address = "me@myisp.com"
objMailMessage.Sender = objMailAddress
```

[C#]

```
AddEmailLib.MailAddressClass objMailAddress = new AddEmailLib.MailAddressClass();
objMailAddress.Address = "me@myisp.com";
objMailMessage.Sender = objMailAddress;
```

[C++]

```
HRESULT get_Sender(IMailAddress**);
HRESULT put_Sender(IMailAddress*);

AddEmailLib::IMailAddressPtr spMailAddress;
spMailAddress.CreateInstance(__uuidof(AddEmailLib::MailAddress));
spMailAddress->PutAddress("me@myisp.com");
spMailMessage->PutSender(spMailAddress);
```

Remarks

Set **Sender** property to the prepared MailAddress object containing e-mail address and optional name of the sender of the message. E-mail address and name will appear in the From field of the message. This property is required and can't be empty.

3.2.3 ReplyTo Property

ReplyTo property specifies e-mail address to reply to.

Syntax

[Visual Basic]

```
Dim objMailAddress As New AddEmailLib.MailAddress
objMailAddress.Address = "me@myisp.com"
objMailMessage.ReplyTo = objMailAddress
```

[VBScript]

```
Dim objMailAddress
Set objMailAddress = CreateObject("AddEmail.MailAddress")
objMailAddress.Address = "me@myisp.com"
objMailMessage.ReplyTo = objMailAddress
```

[C#]

```
AddEmailLib.MailAddressClass objMailAddress = new AddEmailLib.MailAddressClass();
objMailAddress.Address = "me@myisp.com";
objMailMessage.ReplyTo = objMailAddress;
```

[C++]

```
HRESULT get_ReplyTo(IMailAddress**);
HRESULT put_ReplyTo(IMailAddress*);

AddEmailLib::IMailAddressPtr spMailAddress;
spMailAddress.CreateInstance(__uuidof(AddEmailLib::MailAddress));
spMailAddress->PutAddress("me@myisp.com");
spMailMessage->PutReplyTo(spMailAddress);
```

Remarks

Set **ReplyTo** property to the prepared MailAddress object containing e-mail address and optional name of a person who should receive a reply. E-mail address and name will be used when recipient replies to the message. This property is optional and can be empty. If **ReplyTo** is not set e-mail address and name from the Sender property will be used when replying to the message.

3.2.4 MessageSubject Property

MessageSubject property specifies subject of e-mail message.

Syntax

[Visual Basic]

```
objMailMessage.MessageSubject = "test message"
```

[VBScript]

```
objMailMessage.MessageSubject = "test message"
```

[C#]

```
objMailMessage.MessageSubject = "test message";
```

[C++]

```
HRESULT get_MessageSubject(BSTR*);
HRESULT put_MessageSubject(BSTR);

spMailMessage->PutMessageSubject("test message");
```

Remarks

Default value of **MessageSubject** property is an empty string. The message will have no subject if **MessageSubject** property contains an empty string.

3.2.5 MessageBody Property

MessageBody property specifies main body of e-mail message.

Syntax

[Visual Basic]

```
objMailMessage.MessageBody = "Testing, testing, testing..."
```

[VBScript]

```
objMailMessage.MessageBody = "Testing, testing, testing..."
```

[C#]

```
objMailMessage.MessageBody = "Testing, testing, testing...";
```

[C++]

```
HRESULT get_MessageBody(BSTR*);
HRESULT put_MessageBody(BSTR);

spMailMessage->PutMessageBody("Testing, testing, testing...");
```

Remarks

MessageBody can contain plain text or HTML depending on the value of MessageBodyFormat property. Default value of **MessageBody** property is an empty string. The message will have no body if **MessageBody** property contains an empty string.

3.2.6 MessageBodyFormat Property

MessageBodyFormat property specifies format of main body of e-mail message.

Syntax

[Visual Basic]

```
objMailMessage.MessageBodyFormat = MailFormatText
```

[VBScript]

```
objMailMessage.MessageBodyFormat = 0 'MailFormatText
```

[C#]

```
objMailMessage.MessageBodyFormat = AddEmailLib.MailFormat.MailFormatText;
```

[C++]

```
HRESULT get_MessageBodyFormat(MailFormat*);
HRESULT put_MessageBodyFormat(MailFormat);

spMailMessage->PutMessageBodyFormat(AddEmailLib::MailFormatText);
```

Remarks

MessageBodyFormat property indicates format of main body of the message. Possible format values are defined in the MailFormat enumeration. Default value of **MessageBodyFormat** property is **MailFormatText**.

3.2.7 MessageBodyEncoding Property

MessageBodyEncoding property specifies encoding of main body of e-mail message.

Syntax

[Visual Basic]

```
objMailMessage.MessageBodyEncoding = MailEncoding7Bit
```

[VBScript]

```
objMailMessage.MessageBodyEncoding = 1 'MailEncoding7Bit
```

[C#]

```
objMailMessage.MessageBodyEncoding = AddEmailLib.MailEncoding.MailEncoding7Bit;
```

[C++]

```
HRESULT get_MessageBodyEncoding(MailEncoding*);
HRESULT put_MessageBodyEncoding(MailEncoding);
```

```
spMailMessage->PutMessageBodyEncoding(AddEmailLib::MailEncoding7Bit);
```

Remarks

MessageBodyEncoding property indicates which encoding method should be used for main body of the message. Possible encoding values are defined in the MailEncoding enumeration. Default value of **MessageBodyEncoding** property is **MailEncodingDefault**, which means that AddEmail will try to determine best encoding method based on the content of message body. Application can specify encoding method to save time needed to analyze message body. Use **MailEncoding7Bit** if message body contains only English ASCII characters 32-126 and line length does not exceed 76 characters. Use **MailEncoding8Bit** if message body contains non-English ASCII characters and line length does not exceed 76 characters. Use **MailEncodingQuotedPrintable** if line length exceeds 76 characters and formatting should be preserved. Use **MailEncodingBase64** for Unicode messages or if message body contains many non-English ASCII characters. For HTML messages **MailEncodingBase64** or **MailEncodingQuotedPrintable** is recommended.

3.2.8 MessageBodyCharset Property

MessageBodyCharset property specifies character set of main body of e-mail message.

Syntax

[Visual Basic]

```
objMailMessage.MessageBodyCharset = "unicode"
```

[VBScript]

```
objMailMessage.MessageBodyCharset = "unicode"
```

[C#]

```
objMailMessage.MessageBodyCharset = "unicode";
```

[C++]

```
HRESULT get_MessageBodyCharset(BSTR*);
HRESULT put_MessageBodyCharset(BSTR);

spMailMessage->PutMessageBodyCharset("unicode");
```

Remarks

MessageBodyCharset indicates which character set should be used to display main body of the message. Default value of **MessageBodyCharset** property is an empty string. Most common values of **MessageBodyCharset** property are:

"us-ascii" - US ASCII
"iso-8859-1" - Western European (ISO)
"windows-1252" - Western European (Windows)
"unicode" - Unicode (internally converted to UTF-8)

"iso-8859-2" - Central European (ISO)
"windows-1250" - Central European (Windows)

For more information please visit Charsets in Microsoft Internet Explorer.

3.2.9 AltMessageBody Property

AltMessageBody property specifies alternative text-only body of e-mail message.

Syntax

[Visual Basic]

```
objMailMessage.AltMessageBody = "Testing, testing, testing..."
```

[VBScript]

```
objMailMessage.AltMessageBody = "Testing, testing, testing..."
```

[C#]

```
objMailMessage.AltMessageBody = "Testing, testing, testing...";
```

[C++]

```
HRESULT get_AltMessageBody(BSTR*);
HRESULT put_AltMessageBody(BSTR);

spMailMessage->PutAltMessageBody("Testing, testing, testing...");
```

Remarks

AltMessageBody specifies alternative plain-text version of HTML message. **AltMessageBody** will be displayed if e-mail client does not support HTML e-mail messages. Default value of **AltMessageBody** property is an empty string.

3.2.10 AltMessageBodyEncoding Property

AltMessageBodyEncoding property specifies encoding of alternative plain-text body of e-mail message.

Syntax

[Visual Basic]

```
objMailMessage.AltMessageBodyEncoding = MailEncoding7Bit
```

[VBScript]

```
objMailMessage.AltMessageBodyEncoding = 1 'MailEncoding7Bit
```

[C#]

```
objMailMessage.AltMessageBodyEncoding = AddEmailLib.MailEncoding.MailEncoding7Bit;
```

[C++]

```
HRESULT get_AltMessageBodyEncoding(MailEncoding*);
HRESULT put_AltMessageBodyEncoding(MailEncoding);

spMailMessage->PutAltMessageBodyEncoding(AddEmailLib::MailEncoding7Bit);
```

Remarks

AltMessageBodyEncoding property indicates which encoding method should be used for alternative plain-text body of the message. Possible encoding values are defined in the MailEncoding enumeration. Default value of **AltMessageBodyEncoding** property is **MailEncodingDefault**, which means that AddEmail will try to determine best encoding method based on the content of alternative message body. Application can specify encoding method to save time needed to analyze alternative message body. Use **MailEncoding7Bit** if alternative message body contains only English ASCII characters 32-126 and line length does not exceed 76 characters. Use **MailEncoding8Bit** if alternative message body contains non-English ASCII characters and line length does not exceed 76 characters. Use **MailEncodingQuotedPrintable** if line length exceeds 76 characters and formatting should be preserved. Use **MailEncodingBase64** for Unicode messages or if alternative message body contains many non-English ASCII characters.

3.2.11 AltMessageBodyCharset Property

AltMessageBodyCharset property specifies character set of alternative plain-text body of e-mail message.

Syntax

[Visual Basic]

```
objMailMessage.AltMessageBodyCharset = "unicode"
```

[VBScript]

```
objMailMessage.AltMessageBodyCharset = "unicode"
```

[C#]

```
objMailMessage.AltMessageBodyCharset = "unicode";
```

[C++]

```
HRESULT get_AltMessageBodyCharset(BSTR*);
HRESULT put_AltMessageBodyCharset(BSTR);

spMailMessage->PutAltMessageBodyCharset("unicode");
```

Remarks

AltMessageBodyCharset indicates which character set should be used to display alternative body of the message. Default value of **AltMessageBodyCharset** property is an empty string. Most common values of **AltMessageBodyCharset** property are:

"us-ascii" - US ASCII
"iso-8859-1" - Western European (ISO)
"windows-1252" - Western European (Windows)
"unicode" - Unicode (internally converted to UTF-8)
"iso-8859-2" - Central European (ISO)
"windows-1250" - Central European (Windows)

For more information please visit Charsets in Microsoft Internet Explorer.

3.2.12 MessagePriority Property

MessagePriority property specifies priority of e-mail message.

Syntax

[Visual Basic]

```
objMailMessage.MessagePriority = MailPriorityHigh
```

[VBScript]

```
objMailMessage.MessagePriority = 1 'MailPriorityHigh
```

[C#]

```
objMailMessage.MessagePriority = AddEmailLib.MailPriority.MailPriorityHigh;
```

[C++]

```
HRESULT get_MessagePriority(MailPriority*);
HRESULT put_MessagePriority(MailPriority);

spMailMessage->PutMessagePriority(AddEmailLib::MailPriority);
```

Remarks

MessagePriority property indicates priority of the message. Possible priority values are defined in the MailPriority enumeration. Default value of **MessagePriority** property is **MailPriorityNormal**.

3.2.13 MessageId Property

MessageId property specifies ID of e-mail message.

Syntax

[Visual Basic]

```
objMailMessage.MessageId = "123456abc@myisp.com"
```

[VBScript]

```
objMailMessage.MessageId = "123456abc@myisp.com"
```

[C#]

```
objMailMessage.MessageId = "123456abc@myisp.com";
```

[C++]

```
HRESULT get_MessageId(BSTR*);
HRESULT put_MessageId(BSTR);

spMailMessage->PutMessageId("123456abc@myisp.com");
```

Remarks

MessageId is globally unique ID of e-mail message. Default value of **MessageId** property is empty string. In most cases application should not specify ID of the message because it will be assigned by AddEmail or by the outgoing mail server.

3.2.14 MessageBodyFile Property

MessageBodyFile property specifies filename of the file that contains main body of e-mail message.

Syntax

[Visual Basic]

```
objMailMessage.MessageBodyFile = "test.html"
```

[VBScript]

```
objMailMessage.MessageBodyFile = "test.html"
```

[C#]

```
objMailMessage.MessageBodyFile = "test.html";
```

[C++]

```
HRESULT get_MessageBodyFile(BSTR*);
HRESULT put_MessageBodyFile(BSTR);

spMailMessage->PutMessageBodyFile("test.html");
```

Remarks

Instead of supplying message body using **MessageBody** property, application can set **MessageBodyFile** property to the filename of the file that contains the body. AddEmail will read message body from the supplied file. Please note that specified file is not loaded to memory immediately

and must exist until the message is sent.

3.2.15 RequestReadReceipt Property

RequestReadReceipt property specifies whether a read receipt for the message is requested.

Syntax

[Visual Basic]

```
objMailMessage.RequestReadReceipt = True
```

[VBScript]

```
objMailMessage.RequestReadReceipt = True
```

[C#]

```
objMailMessage.RequestReadReceipt = true;
```

[C++]

```
HRESULT get_RequestReadReceipt(VARIANT_BOOL*);
HRESULT put_RequestReadReceipt(VARIANT_BOOL);

spMailMessage->PutRequestReadReceipt(VARIANT_TRUE);
```

Remarks

Set **RequestReadReceipt** to *True* if you want to request a read receipt for the message. Default value of **RequestReadReceipt** property is *False*.

3.2.16 AddRecipient Method

AddRecipient method adds recipient to the To field of e-mail message.

Syntax

[Visual Basic]

```
Sub AddRecipient(objMailAddress As AddEmailLib.MailAddress)
```

Example:

```
Dim objMailAddress As New AddEmailLib.MailAddress
objMailAddress.Address = "someone@someisp.com"
objMailMessage.AddRecipient objMailAddress
```

[VBScript]

```
Sub AddRecipient(objMailAddress)
```

Example:

```
Dim objMailAddress
Set objMailAddress = CreateObject("AddEmail.MailAddress")
objMailAddress.Address = "someone@someisp.com"
objMailMessage.AddRecipient objMailAddress
```

[C#]

```
void AddRecipient(AddEmailLib.MailAddressClass objMailAddress);
```

Example:

```
AddEmailLib.MailAddressClass objMailAddress = new AddEmailLib.MailAddressClass();
objMailAddress.Address = "someone@somesp.com";
objMailMessage.AddRecipient(objMailAddress);
```

[C++]

```
HRESULT AddRecipient(IMailAddress* objMailAddress);
```

Example:

```
AddEmailLib::IMailAddressPtr spMailAddress;
spMailAddress.CreateInstance(__uuidof(AddEmailLib::MailAddress));
spMailAddress->PutAddress("someone@someisp.com");
spMailMessage->AddRecipient(spMailAddress);
```

Parameters

`objMailAddress` *[in]*
Prepared MailAddress object containing e-mail address and name of the recipient.

Return value

None.

Remarks

Use **AddRecipient** method to add prepared MailAddress object containing e-mail address and optional name of the recipient of e-mail message. E-mail address and name will appear in the To field of the message. Application can call **AddRecipient** method more than once to have more than one recipient in the To field.

3.2.17 AddCcRecipient Method

AddCcRecipient method adds recipient to the Cc field of e-mail message.

Syntax

[Visual Basic]

```
Sub AddCcRecipient(objMailAddress As AddEmailLib.MailAddress)
```

Example:

```
Dim objMailAddress As New AddEmailLib.MailAddress
objMailAddress.Address = "someone@someisp.com"
objMailMessage.AddCcRecipient objMailAddress
```

[VBScript]

```
Sub AddCcRecipient(objMailAddress)
```

Example:

```
Dim objMailAddress
Set objMailAddress = CreateObject("AddEmail.MailAddress")
objMailAddress.Address = "someone@someisp.com"
objMailMessage.AddCcRecipient objMailAddress
```

[C#]

```
void AddCcRecipient(AddEmailLib.MailAddressClass objMailAddress);
```

Example:

```
AddEmailLib.MailAddressClass objMailAddress = new AddEmailLib.MailAddressClass();
objMailAddress.Address = "someone@somesp.com";
objMailMessage.AddCcRecipient(objMailAddress);
```

[C++]

```
HRESULT AddCcRecipient(IMailAddress* objMailAddress);
```

Example:

```
AddEmailLib::IMailAddressPtr spMailAddress;
spMailAddress.CreateInstance(__uuidof(AddEmailLib::MailAddress));
spMailAddress->PutAddress("someone@someisp.com");
spMailMessage->AddCcRecipient(spMailAddress);
```

Parameters

`objMailAddress` *[in]*
Prepared MailAddress object containing e-mail address and name of the recipient.

Return value

None.

Remarks

Use **AddCcRecipient** method to add prepared MailAddress object containing e-mail address and optional name of the recipient of e-mail message. E-mail address and name will appear in the Cc field of the message. Application can call **AddCcRecipient** method more than once to have more than one

recipient in the Cc field.

3.2.18 AddBccRecipient Method

AddBccRecipient method adds Bcc (invisible) recipient to e-mail message.

Syntax

[Visual Basic]

```
Sub AddBccRecipient(objMailAddress As AddEmailLib.MailAddress)
```

Example:

```
Dim objMailAddress As New AddEmailLib.MailAddress
objMailAddress.Address = "someone@someisp.com"
objMailMessage.AddBccRecipient objMailAddress
```

[VBScript]

```
Sub AddBccRecipient(objMailAddress)
```

Example:

```
Dim objMailAddress
Set objMailAddress = CreateObject("AddEmail.MailAddress")
objMailAddress.Address = "someone@someisp.com"
objMailMessage.AddBccRecipient objMailAddress
```

[C#]

```
void AddBccRecipient(AddEmailLib.MailAddressClass objMailAddress);
```

Example:

```
AddEmailLib.MailAddressClass objMailAddress = new AddEmailLib.MailAddressClass();
objMailAddress.Address = "someone@somesp.com";
objMailMessage.AddBccRecipient(objMailAddress);
```

[C++]

```
HRESULT AddBccRecipient(IMailAddress* objMailAddress);
```

Example:

```
AddEmailLib::IMailAddressPtr spMailAddress;
spMailAddress.CreateInstance(__uuidof(AddEmailLib::MailAddress));
spMailAddress->PutAddress("someone@someisp.com");
spMailMessage->AddBccRecipient(spMailAddress);
```

Parameters

`objMailAddress` *[in]*

Prepared MailAddress object containing e-mail address and name of the recipient.

Return value

None.

Remarks

Use **AddBccRecipient** method to add prepared MailAddress object containing e-mail address and optional name of the recipient of e-mail message. E-mail address and name will not visible, but the message will be sent to Bcc recipients. Application can call **AddBccRecipient** method more than once to specify more than one Bcc recipient.

3.2.19 AddAttachment Method

AddAttachment method adds attachment to e-mail message.

Syntax

[Visual Basic]

```
Sub AddAttachment(objMailAttachment As AddEmailLib.MailAttachment)
```

Example:

```
Dim objMailAttachment As New AddEmailLib.MailAttachment
objMailAttachment.File = "c:\files\document.pdf"
objMailMessage.AddAttachment objMailAttachment
```

[VBScript]

```
Sub AddAttachment(objMailAttachment)
```

Example:

```
Dim objMailAttachment
Set objMailAttachment = CreateObject("AddEmail.MailAttachment")
objMailAttachment.File = "c:\files\document.pdf"
objMailMessage.AddAttachment objMailAttachment
```

[C#]

```
void AddAttachment(AddEmailLib.MailAttachmentClass objMailAttachment);
```

Example:

```
AddEmailLib.MailAttachmentClass objMailAttachment = new
AddEmailLib.MailAttachmentClass();
objMailAttachment.File = "c:\\files\\document.pdf";
objMailMessage.AddAttachment(objMailAttachment);
```

[C++]

```
HRESULT AddAttachment(IMailAddress* objMailAttachment);
```

Example:

```
AddEmailLib::IMailAttachmentPtr spMailAttachment;
spMailAttachment.CreateInstance(__uuidof(AddEmailLib::MailAttachment));
spMailAttachment->PutFile("c:\\files\\document.pdf");
spMailMessage->AddAttachment(spMailAttachment);
```

Parameters

`objMailAttachment` *[in]*
Prepared MailAttachment object containing attachment.

Return value

None.

Remarks

Use **AddAttachment** method to add prepared MailAttachment object to the e-mail message. Application can call **AddAttachment** method more than once to add more than one attachment to the message.

3.2.20 AddHeader Method

AddHeader method adds header value to e-mail message.

Syntax

[Visual Basic]

```
Sub AddHeader(strHeader As String, strValue As String)
```

Example:

```
objMailMessage.AddHeader "X-Mailer", "My Email Application 1.0"
```

[VBScript]

```
Sub AddHeader(strHeader, strValue)
```

Example:

```
objMailMessage.AddHeader "X-Mailer", "My Email Application 1.0"
```

[C#]

```
void AddHeader(string strHeader, string strValue);
```

Example:

```
objMailMessage.AddHeader( "X-Mailer", "My Email Application 1.0" );
```

[C++]

```
HRESULT AddHeader(BSTR strHeader, BSTR strValue);
```

Example:

```
spMailMessage->AddHeader("X-Mailer", "My Email Application 1.0");
```

Parameters

`strHeader` *[in]*
String.

`strValue` *[in]*
String.

Return value

None.

Remarks

Use **AddHeader** method to add custom header and its value to the e-mail message. Most common header is "X-Mailer", it contains name of an application that sent an e-mail.

3.2.21 ClearAttachments Method

ClearAttachments method clears list of attachments of e-mail message.

Syntax

[Visual Basic]

```
Sub ClearAttachments()
```

Example:

```
objMailMessage.ClearAttachments
```

[VBScript]

```
Sub ClearAttachments()
```

Example:

```
objMailMessage.ClearAttachments
```

[C#]

```
void ClearAttachments();
```

Example:

```
objMailMessage.ClearAttachments();
```

[C++]

```
HRESULT ClearAttachments();
```

Example:

```
spMailMessage->ClearAttachments();
```

Parameters

None.

Return value

None.

Remarks

Use **ClearAttachments** method to remove all attachments from e-mail message.

3.2.22 ClearRecipients Method

ClearRecipients method clears list of recipients in To field of e-mail message.

Syntax

[Visual Basic]

```
Sub ClearRecipients()
```

Example:

```
objMailMessage.ClearRecipients
```

[VBScript]

```
Sub ClearRecipients()
```

Example:

```
objMailMessage.ClearRecipients
```

[C#]

```
void ClearRecipients();
```

Example:

```
objMailMessage.ClearRecipients();
```

[C++]

```
HRESULT ClearRecipients();
```

Example:

```
spMailMessage->ClearRecipients();
```

Parameters

None.

Return value

None.

Remarks

Use **ClearRecipients** method to remove all recipients from To field of e-mail message. Using this method you can clear list of recipients after sending and reuse prepared MailMessage object to send the same message to other recipients.

3.2.23 ClearCcRecipients Method

ClearCcRecipients method clears list of recipients in Cc field of e-mail message.

Syntax

[Visual Basic]

```
Sub ClearCcRecipients()
```

Example:

```
objMailMessage.ClearCcRecipients
```

[VBScript]

```
Sub ClearCcRecipients()
```

Example:

```
objMailMessage.ClearCcRecipients
```

[C#]

```
void ClearCcRecipients();
```

Example:

```
objMailMessage.ClearCcRecipients();
```

[C++]

```
HRESULT ClearCcRecipients();
```

Example:

```
spMailMessage->ClearCcRecipients();
```

Parameters

None.

Return value

None.

Remarks

Use **ClearCcRecipients** method to remove all recipients from Cc field of e-mail message. Using this method you can clear list of recipients after sending and reuse prepared MailMessage object to send the same message to other recipients.

3.2.24 ClearBccRecipients Method

ClearBccRecipients method clears list of recipients in Bcc field of e-mail message.

Syntax

[Visual Basic]

```
Sub ClearBccRecipients()
```

Example:

```
objMailMessage.ClearBccRecipients
```

[VBScript]

```
Sub ClearBccRecipients()
```

Example:

```
objMailMessage.ClearBccRecipients
```

[C#]

```
void ClearBccRecipients();
```

Example:

```
objMailMessage.ClearBccRecipients();
```

[C++]

```
HRESULT ClearBccRecipients();
```

Example:

```
spMailMessage->ClearBccRecipients();
```

Parameters

None.

Return value

None.

Remarks

Use **ClearBccRecipients** method to remove all recipients from Bcc field of e-mail message. Using this method you can clear list of recipients after sending and reuse prepared MailMessage object to send the same message to other recipients.

3.2.25 ImportHTML Method

ImportHTML method imports HTML file with embedded images.

Syntax

[Visual Basic]

```
Sub ImportHTML(strFilename As String)
```

Example:

```
objMailMessage.ImportHTML "c:\files\email.html"
```

[VBScript]

```
Sub ImportHTML(strFilename)
```

Example:

```
objMailMessage.ImportHTML "c:\files\email.html"
```

[C#]

```
void ImportHTML(string strFilename);
```

Example:

```
objMailMessage.ImportHTML( "c:\\files\\email.html" );
```

[C++]

```
HRESULT ImportHTML(BSTR strFilename);
```

Example:

```
spMailMessage->ImportHTML("c:\\files\\email.html");
```

Parameters

`strFilename` *[in]*
Filename of the HTML file to be imported.

Return value

None.

Remarks

Use **ImportHTML** method to create HTML e-mail with embedded images from a file on disk. This method imports specified HTML file, adds all images referenced in the HTML as inline attachments, modifies HTML as needed and fills out the MessageBody. Only images located on disk are added as attachments. Images from the web (http:// or https:// links in HTML) are not attached. Please note that the current version of AddEmail does not import images specified in CSS using url values.

3.3 MailAttachment

3.3.1 Overview

MailAttachment object represents message attachment.

Syntax:

[Visual Basic]

```
Dim objMailAttachment As New AddEmailLib.MailAttachment
objMailAttachment.File = "c:\files\document.pdf"
```

[VBScript]

```
Dim objMailAttachment
Set objMailAttachment = CreateObject("AddEmail.MailAttachment")
objMailAttachment.File = "c:\files\document.pdf"
```

[C#]

```
AddEmailLib.MailAttachmentClass objMailAttachment = new
AddEmailLib.MailAttachmentClass();
objMailAttachment.File = "c:\\files\\document.pdf";
```

[C++]

```
AddEmailLib::IMailAttachmentPtr spMailAttachment;
spMailAttachment.CreateInstance(__uuidof(AddEmailLib::MailAttachment));
spMailAttachment->PutFile( "c:\\files\\document.pdf" );
```

Properties:

File	Filename of the attachment file.
Name	Name of the attachment as it should appear in e-mail message.
Encoding	Encoding of the attachment.
ContentType	Content type of the attachment.
Charset	Character set of the attachment.
Inline	Specifies whether the attachment is inline, e.g. embedded in e-mail message.
ContentId	Content Id of the attachment.

Methods:

LoadFromMemory	Loads attachment data from memory instead of file.

3.3.2 File Property

File property specifies filename (including path) of the attachment file.

Syntax

[Visual Basic]

```
objMailAttachment.File = "c:\files\document.pdf"
```

[VBScript]

```
objMailAttachment.File = "c:\files\document.pdf"
```

[C#]

```
objMailAttachment.File = "c:\\files\\document.pdf";
```

[C++]

```
HRESULT get_File(BSTR*);
HRESULT put_File(BSTR);

spMailAttachment->PutFile("c:\\files\\document.pdf");
```

Remarks

Set **File** property to the full filename of the attachment. AddEmail will read attachment from the specified

file. Please note that specified file is not loaded to memory immediately and must exist until the message is sent.

3.3.3 Name Property

Name property specifies name of the attachment as it should appear in the e-mail.

Syntax

[Visual Basic]

```
objMailAttachment.Name = "Report 2004.Pdf"
```

[VBScript]

```
objMailAttachment.Name = "Report 2004.Pdf"
```

[C#]

```
objMailAttachment.Name = "Report 2004.Pdf";
```

[C++]

```
HRESULT get_Name(BSTR*);
HRESULT put_Name(BSTR);

spMailAttachment->PutName( "Report 2004.Pdf" );
```

Remarks

Default value of **Name** property is an empty string. If **Name** is an empty string, name part of the filename specified in the File property is used.

3.3.4 Encoding Property

Encoding property specifies encoding of the attachment.

Syntax

[Visual Basic]

```
Dim attachmentEncoding As AddEmailLib.MailEncoding
objMailAttachment.Encoding = MailEncoding7Bit
```

[VBScript]

```
Dim attachmentEncoding
objMailAttachment.Encoding = 1 'MailEncoding7Bit
```

[C#]

```
objMailAttachment.Encoding = AddEmailLib.MailEncoding.MailEncoding7Bit;
```

[C++]

```
HRESULT get_Encoding(MailEncoding*);
HRESULT put_Encoding(MailEncoding);

spMailAttachment->PutEncoding(AddEmailLib::MailEncoding7Bit);
```

Remarks

Encoding property indicates which encoding method should be used for the attachment. Possible encoding values are defined in the MailEncoding enumeration. Default value of **Encoding** property is **MailEncodingBase64** and usually doesn't have to be changed. Do not use encoding other than **MailEncodingBase64** for binary files.

3.3.5 ContentType Property

ContentType property specifies content type of the attachment.

Syntax

[Visual Basic]

```
objMailAttachment.ContentType = "application/pdf"
```

[VBScript]

```
objMailAttachment.ContentType = "application/pdf"
```

[C#]

```
objMailAttachment.ContentType = "application/pdf";
```

[C++]

```
HRESULT get_ContentType(BSTR*);
HRESULT put_ContentType(BSTR);

spMailAttachment->PutContentType("application/pdf");
```

Remarks

Default value of **ContentType** property is an empty string. If **ContentType** is an empty string, AddEmail tries to read content type from the registry based on file extension of the attachment.

3.3.6 Charset Property

Charset property specifies character set of the attachment.

Syntax

[Visual Basic]

```
objMailAttachment.Charset = "iso-8859-1"
```

[VBScript]

```
objMailAttachment.Charset = "iso-8859-1"
```

[C#]

```
objMailAttachment.Charset = "iso-8859-1";
```

[C++]

```
HRESULT get_Charset(BSTR*);
HRESULT put_Charset(BSTR);

spMailAttachment->PutCharset("iso-8859-1");
```

Remarks

Charset indicates which character set should be used to display the attachment. Default value of **Charset** property is an empty string. Only text attachments should have **Charset** property set.

3.3.7 Inline Property

Inline property specifies whether the attachment is inline (embedded in the message).

Syntax

[Visual Basic]

```
objMailAttachment.Inline = True
```

[VBScript]

```
objMailAttachment.Inline = True
```

[C#]

```
objMailAttachment.Inline = true;
```

[C++]

```
HRESULT get_Inline(VARIANT_BOOL*);
HRESULT put_Inline(VARIANT_BOOL);

spMailAttachment->PutInline(VARIANT_TRUE);
```

Remarks

Default value of **Inline** property is *False*. Set **Inline** property to *True* for embedded images. Please refer

to the **EmbeddedImages** sample for details.

3.3.8 ContentId Property

ContentId property specifies ID of the embedded attachment. ID is used to reference attachment in HTML body of e-mail message.

Syntax

[Visual Basic]

```
objMailAttachment.ContentId = "Img1"
```

[VBScript]

```
objMailAttachment.ContentId = "Img1"
```

[C#]

```
objMailAttachment.ContentId = "Img1";
```

[C++]

```
HRESULT get_ContentId(BSTR*);
HRESULT put_ContentId(BSTR);

spMailAttachment->PutContentId("Img1");
```

Remarks

Default value of **ContentId** property is an empty string. Application should set **ContentId** of the embedded attachments to be able to reference them in the HTML body of e-mail message. Please refer to the **EmbeddedImages** sample for details.

3.3.9 LoadFromMemory Method

LoadFromMemory method loads attachment data from memory instead of file.

Syntax

[C++]

```
HRESULT LoadFromMemory(BYTE* data, LONG size);
```

Example:

```
BYTE data[100];
// fill data array
spMailAttachment->LoadFromMemory(data, 100);
spMailAttachment->PutName("document.pdf");
```

Parameters

`data` *[in]*
BYTE array with attachment data.

`size` *[in]*
Size of attachment data.

Return value

None.

Remarks

Use **LoadFromMemory** method to create in-memory attachment. File property is not used in this case. Application should set Name property to specify name of the attachment displayed in the e-mail.

3.4 MailAddress

3.4.1 Overview

MailAddress object represents e-mail address.

Syntax:

[Visual Basic]

```
Dim objMailAddress As New AddEmailLib.MailAddress
objMailAddress.Address = "someone@someisp.com"
```

[VBScript]

```
Dim objMailAddress
Set objMailAddress = CreateObject("AddEmail.MailAddress")
objMailAddress.Address = "someone@someisp.com"
```

[C#]

```
AddEmailLib.MailAddressClass objMailAddress = new AddEmailLib.MailAddressClass();
objMailAddress.Address = "someone@someisp.com";
```

[C++]

```
AddEmailLib::IMailAddressPtr spMailAddress;
spMailAddress.CreateInstance(__uuidof(AddEmailLib::MailAddress));
spMailAddress->PutAddress("someone@someisp.com");
```

Properties:

Address	E-mail address.
Name	Name as it should appear in e-mail message.

3.4.2 Address Property

Address property specifies e-mail address of recipient or sender.

Syntax

[Visual Basic]

```
objMailAddress.Address = "smithj@someisp.com"
```

[VBScript]

```
objMailAddress.Address = "smithj@someisp.com"
```

[C#]

```
objMailAddress.Address = "smithj@someisp.com";
```

[C++]

```
HRESULT get_Address(BSTR*);
HRESULT put_Address(BSTR);

spMailAddress->PutAddress("smithj@someisp.com");
```

Remarks

3.4.3 Name Property

Name property specifies name of recipient or sender as it should appear in the e-mail.

Syntax

[Visual Basic]

```
objMailAddress.Name = "John Smith"
```

[VBScript]

```
objMailAddress.Name = "John Smith"
```

[C#]

```
objMailAddress.Name = "John Smith";
```

[C++]

```
HRESULT get_Name(BSTR*);
HRESULT put_Name(BSTR);
```

```
spMailAddress->PutName("John Smith");
```

Remarks

If **Name** property is an empty string, only e-mail address will be displayed in the e-mail message.

3.5 MailFormat

The following table describes possible values of **MailFormat** enumeration.

Value	Numeric value	Description
MailFormatText	0	Specifies text format.
MailFormatHtml	1	Specifies HTML format.

3.6 MailEncoding

The following table describes possible values of **MailEncoding** enumeration.

Value	Numeric value	Description
MailEncodingDefault	0	AddEmail will select suitable encoding based on the content.
MailEncoding7Bit	1	7-bit encoding suitable for English ASCII text with line length 76 characters or less.
MailEncoding8Bit	2	8-bit encoding suitable for mostly English ASCII text with some non-English characters with line length 76 characters or less.
MailEncodingQuotedPrintable	3	Quoted-printable encoding suitable for English or non-English ASCII text with any line length. Preserves line formatting.
MailEncodingBase64	4	Base64 encoding suitable for binary files, Unicode text or non-English text.

3.7 MailPriority

The following table describes possible values of **MailPriority** enumeration.

Value	Numeric value	Description
MailPriorityHigh	1	Specifies higher priority (urgent) message.
MailPriorityNormal	3	Specifies normal priority message.
MailPriorityLow	5	Specifies lower priority message.

3.8 MailStatus

The following table describes possible values of **MailStatus** enumeration.

Value	Numeric value	Description
MailStatusQueued	0	Message was added to the queue but AddEmail didn't start sending it yet.
MailStatusSending	1	AddEmail is currently sending this message.
MailStatusSent	2	Message was sent successfully.
MailStatusFailed	3	AddEmail was unable to send this message.
MailStatusCanceled	4	Message was canceled using Cancel or CancelAll method.