



Messaging Services for Microsoft Windows

MailEnable API Guide

Issue Date: 11th August 2009

Table of Contents

1	Introduction	3
2	MailEnable Constructs	4
2.1	Address Formatting	4
2.2	Message Command Files.....	5
3	MailEnable COM Administration Objects	8
3.1	Address Map Administration	9
3.2	Authentication Administration.....	11
3.3	List Server Administration	12
3.4	Post Office Administration.....	16
3.5	POP Administration.....	20
3.6	SMTP Administration.....	22
3.7	Remote Administration	28
3.7.1	Registering the Host	28
3.7.2	Registering a Session for Remote Administration	28
3.7.3	Calling the Remote Object.....	29
3.7.4	Accessing Miscellaneous Configuration Settings	29
3.8	Server Management Groups and Clusters	30
4	Developing Connectors.....	31
4.1	Implementing a Connector	31
4.2	Considerations.....	32
5	Developing MTA Pickup Events.....	33
5.1	Developing a Pickup Event Application	33
5.2	Developing a Pickup Event Class	34
5.2.1	Developing the Object.....	34
5.2.2	Integrating the COM Pickup Event.....	35
6	Developing Mailbox Delivery Events.....	37
7	SendMail COM Component	38
8	Developing Skins	41
8.1	Developing Skins	41
8.1.1	Default Skins.....	41
8.1.2	Enterprise Skins.....	Error! Bookmark not defined.
8.2	Skins – Additional Considerations	47
8.3	Creating a Skin.....	48
8.4	Distributing Skins	48
8.5	Developing Base Applications.....	48
8.6	Licensing Considerations	48
9	Examples	49
9.1	Creating a Post Office.....	49
9.2	Deleting a Post Office.....	50
10	More Information	52

1 Introduction

MailEnable exposes an array of interfaces that you can use to automate its behaviour or extend the application. This API Guide explains these interfaces and provides examples of how to configure MailEnable programmatically.

IMPORTANT: You are not permitted to distribute MailEnable components in any product or extension. The target environment for your application or extension must either have the Standard, Professional or Enterprise Edition of MailEnable installed.

MailEnable takes reasonable effort to ensure that the information provided in this API guide is accurate at the time of publishing. From time to time, the API may change as extensions emerge and such changes will be communicated through revisions to the API guide.

2 MailEnable Constructs

This section outlines some basic construction and underlying principals with respect to the inner workings of MailEnable.

2.1 Address Formatting

An internal MailEnable address is made of two core parts. Firstly, there is the Connector Descriptor and secondly there is the addressing detail. The exact syntax is shown below:

Syntax:

[Connector Acronym: Connector Address Details]

Examples:

MailEnable Internal Address	Explanation
[SF:POSTOFFICE/MAILBOX]	The location of mailbox (MAILBOX) on postoffice (POSTOFFICE) using the SF Connector (Postoffice Connector)
[SMTP:User@domain]	The SMTP Address of a user at the prescribed domain using the SMTP Connector
[LS:POSTOFFICE/LISTNAME]	The location of list mailbox (LISTNAME) on postoffice (POSTOFFICE) using the LS Connector (List Connector).

When a mail connector receives mail, it resolves the addressed recipients to an internal address format. Some of these recipients will be local, and others will be relayed to non-local users. The connector will produce a command file containing all resolved recipients and a message file containing the actual data. This information is stored in the Connectors Spooling directory.

2.2 Message Command Files

For most messages queues there are two files per message. The first is called a **command file** and the second is the message body. The command file contains details concerning the delivery of the message (such as sender, retry count, etc.). These details may vary slightly depending on the queue the message is in. For example, the command file for the SMTP connector is located in the Mail Enable\Queues\SMTP\Outgoing directory. It has the file extension .MAI. The first part of the name should be a GUID to avoid overwriting existing files. The command file corresponds to the actual message that is located in the Mail Enable\Queues\SMTP\Outgoing\Messages directory. Make sure you write the message file before you write the command file, in order to avoid the SMTP service reading the command file and not finding the message.

The command file has the following format:

```
DomainName=[Setting]<CRLF>
CommandType=[Setting]<CRLF>
Server=[Setting]<CRLF>
Recipients=[Setting]<CRLF>
Sender=[Setting]<CRLF>
Retries=[Setting]<CRLF>
NextSendTime=[Setting]<CRLF>
TimeAcquired=[Setting]<CRLF>
MessageID=[Setting]<CRLF>
Priority=[Setting]<CRLF>
Status=[Setting]
IPAddress=[Setting]<CRLF>
Account=[Setting]<CRLF>
AuthenticationStatus=[Setting]<CRLF>
Subject=[Setting]
```

The values of the items are described below:

DomainName:	This is the name of the domain that will be connected to. If this is a new message, you can leave this blank (i.e. just have DomainName=), as MailEnable will resolve it. This is only used only when sending messages out the SMTP Connector and is written only by the connector once MX lookup has occurred.
CommandType:	Indicates whether the message is a non-delivery receipt. For new messages this should be "Normal", and for system generated messages it is "NDR". When NDR messages are sent outbound from the SMTP connector they are sent with a NULL sender address (<>).
Server:	The name of the Windows server that is to process the file. You do not need to set this at this stage. You do not need to set this value unless you wish to denote the message as being owned by a particular server (ie: if you are clustering).

Recipients:	<p>The email address of the recipients. You can one or more recipients. If you have more than one recipient, separate the addresses with a semi-colon, such as:</p> <p>[SMTP:email@domain.com]; [SMTP:anotheremail@domain.com]</p> <p>Keep the total recipient length to be less than 24000 bytes.</p>
Sender:	The email address of the sender. This is formatted using the MailEnable address format.
Retries:	The amount of retries the message has had. For new messages set this to zero.
NextSendTime:	The message will tried to be sent if the current time is greater than this time. This is measured in the number of seconds since midnight January 1, 1970, in local time. To send a message immediately, set this to the same value as TimeAcquired.
TimeAquired:	The time the message was created. This is used when checking to see if the message has expired. This is measured in the number of seconds since midnight January 1, 1970, in local time.
MessageID:	The name of the message file. Both the message and command files should have the same filename.
Priority:	Unused. Usually set to "Normal".
Status:	What status the message is in. Set this to "Unsent" for new messages. When a message is being sent, this becomes "Sending".
IPAddress:	This is the IP address of the originating client (it is only relevant/populated by SMTP Inbound).
Account:	This field denotes the 'Owner' Account or Postoffice of message. This records the Account/Postoffice who would get billed for the message (by an integrated billing system).
AuthenticationStatus:	<p>This field indicates whether the sender of the message has authenticated before inserting the message into the MailEnable Queues. In the case of SMTP Inbound messages, this entry is populated with a value other than zero under the following circumstances:</p> <ul style="list-style-type: none"> • The sender authenticated via SMTP Authentication, sets value to "1" • The sender has been permitted to relay from an authorized IP address range – sets value to "2". <p>Note: the authentication flag value of "1" will take precedence over the value of "2". Ie: If the sender has authenticated and are authorized to relay, the value will be set to "1".</p>
Subject	The subject of the message

Example command file:

```
DomainName=mailenable.com
CommandType=NORMAL
Server=
Recipients=[SMTP:info@mailenable.com]
Sender=[SMTP:support@mailenable.com]
Retries=2
NextSendTime=1009515032
TimeAcquired=1009407032
MessageID=D9880414C29A4DEC94C02457718EE.MAI
Priority=Normal
Status=Unsent
Server=MESRV01
```

3 MailEnable COM Administration Objects

MailEnable Standard comes with COM components to allow you to fully control the MailEnable configuration from within your own program or environment. These components provide the interface to the provider DLLs. If your development environment can utilise COM components (such as ASP, Visual Basic, Visual C++, etc.) you can leverage all the administration functions available. By using the components provided, you ensure that your program will maintain compatibility. For instance, using these functions will work whether you are using the default provider of Tab Delimited Files or the ODBC provider.

The files that you would use to develop with are below. Head to the relevant chapters in this document to find all the functions that they expose. All the components you would use begin with the letters MEAO, which is an acronym for MailEnable Administration Object.

Library	Purpose	Versions
MEAOAM	Contains address map administration functions	All
MEAOAU	Contains authentication functions	All
MEAODP	Contains directory administration functions	Enterprise
MEAOLS	Contains List Server administration functions	All
MEAOPO	Contains Post Office administration functions	All
MEAOPS	Contains POP administration functions	All
MEAOSM	Contains SMTP administration functions	All
MEAOSO	Contains System option and configuration functions	All

All the components are objects. Therefore, in order to use, you must instantiate them, set the properties and perform the required functions.

When using the MailEnable COM Interface, you should consider the following:

- Remember that all addresses for the providers use the MailEnable address format, which is described on the next page.
- You don't need to specify all the properties for a class. So if you wish to iterate through all the items in the class (by using FindNext), then just assign an empty string to the string properties, and a -1 to the long values. But not all properties can be used for this pattern matching, so check with the class description to see what you can use. If you are using a Get function, you will get the first item that matches your specifications. If using an edit function, all the items that match your criteria will be affected. When you develop your application, be very careful about the wildcards you use, because if you pass an empty string you may erase all the information in a file. It is best to check for empty strings before performing these actions.

- When there is a list of addresses that can be used, separate the addresses with a comma. For example, if specifying multiple addresses to redirect a mailbox to, you could use the following:

[\[SMTP:peter@mailenable.net\],\[SMTP:peter@mailenable.info\]](#)

- Some items you are trying to add may require multiple commands. For instance, if you wish to add a new mailbox and have an email address for it, you would have to perform the following tasks:
 1. Create a mailbox
 2. Create an address map
 3. Create a login
- Don't use the ClassID to bind your application to the DLLs. Always use the ProgID. Otherwise additions and changes to the DLL interface may cause compatibility problems in the future.

3.1 Address Map Administration

Class

MEAOAM.AddressMap

Properties

Wildcard	Name	Type	Description
Yes	Account	String(1024)	The account/post office
Yes	SourceAddress	String(1024)	The address the email was sent to
Yes	DestinationAddress	String(1024)	The address to send the email to
Yes	Scope	String(1024)	Not used
Yes	Status	Long	0=Enabled, 1=Enabled, 2=Disabled

Functions

GetAddressMap() As Long

FindFirstAddressMap() As Long

FindNextAddressMap() As Long

AddAddressMap() As Long

RemoveAddressMap() As Long

EditAddressMap(ByVal NewAccount As String, ByVal NewSourceAddress As String, ByVal NewDestinationAddress As String, ByVal NewScope As String) As Long

Remarks

Functions return a value of 1 for success and 0 for failure. Other status codes may be returned as information on errors. The AddressMap class is used to direct the incoming mail to the correct connector. Connectors would add their own entries to this file, as the MTA uses this to determine which connector is responsible for it.

Catch-all addresses are handled in MailEnable by the use of a wildcard in the email local part, for example [*@example.com](#). Wildcards have be treated carefully when using the API, as they are used as wildcards, so when deleting a catchall address map it has to be renamed before removal, since removing a catchall address with * in it will remove all addresses for the domain. So before removing do an Edit to rename the email address, then remove this renamed entry.

Example

```
Dim lResult, oAddressMap

Set oAddressMap = CreateObject("MEAOAM.AddressMap")

oAddressMap.Account = ""
oAddressMap.DestinationAddress = ""
oAddressMap.Scope = ""
oAddressMap.SourceAddress = "[SMTP:test@mailenable.com]"

lResult = oAddressMap.GetAddressMap()

If lResult = 0 Then
    MsgBox "Failed to get address."
Else
    MsgBox "Address is sent to: " & oAddressMap.DestinationAddress
End If
```

3.2 Authentication Administration

Class

MEAOAU.Login

Properties

Wildcard	Name	Type	Description
Yes	Username	String(64)	Username
Yes	Status	Long	0=Disabled, 1=Enabled
Yes	Password	String(64)	Password
Yes	Account	String(128)	Account/post office
Yes	Rights	Rights(128)	Unused
Yes	Description	Description(1024)	Unused
	LoginAttempts	Long	Unused
	LastAttempt	Long	Unused
	LastSuccessfulLogin	Long	Unused

Functions

GetLogin() As Long

FindFirstLogin() As Long

FindNextLogin() As Long

AddLogin() As Long

RemoveLogin() As Long

EditLogin(ByVal NewUserName As String, ByVal NewStatus As Long, ByVal NewPassword As String, ByVal NewAccount As String, ByVal NewDescription As String, ByVal NewLoginAttempts As Long, ByVal NewLastAttempt As Long, ByVal NewLastSuccessfulLogin As Long, ByVal NewRights As String) As Long

Remarks

Functions return non-zero for success, zero for failure. The authentication class is used to authenticate a users username and password combination. It can, and is, used for a variety of services and connectors. For example, the POP service would use it to validate a user logon.

Use only 30 characters maximum for the password. This is because encrypted passwords take up over twice as many characters, even though the provider will always return the unencrypted passwords. Remember that the encryption key in the registry must be correct, or the password returned will be wrong when using encrypted passwords.

You are able to pattern match on the following properties:

- Username
- Status
- Password
- Account
- Rights
- Description

3.3 Directory Administration

Class

MEAODP.Directory

Properties

Wildcard	Name	Type	Description
	DirectoryEntryID	String(256)	
	DisplayName	String(256)	
	Account	String(128)	Postoffice
	MailAddress	String(1024)	
	DirectoryLocatorID	String(2048)	
	EntryType	Long	
	Host	String(128)	

3.4 List Server Administration

Class

MEAOLS.List

Properties

Wildcard	Name	Type	Description
	Description	String(256)	List description
	AccountName	String(128)	Account/post office
Yes	ListName	String(128)	Name of list
	ListType	Long	0=Unmoderated, 1=Moderated
	ListStatus	Long	0=Disabled, 1=Enabled
	ModeratorAddress	String(128)	Moderator address
	HeaderAnnotationStatus	Long	0=no header, 1=Include header file
	HeaderAnnotation	String(256)	Name of header file with no extension
	FooterAnnotationStatus	Long	0=no footer, 1=Include footer file
	FooterAnnotation	String(256)	Name of footer file with no extension
	ListAddress	String(256)	Address of list
(Reserved)	SubscribeMessageFileStatus	Long	
(Reserved)	SubscribeMessageFile	String(256)	
(Reserved)	UnsubscribeMessageFileStatus	Long	
(Reserved)	UnsubscribeMessageFile	String(256)	
(Reserved)	SubjectSuffixStatus	Long	
(Reserved)	SubjectSuffix	String(256)	
	SubjectPrefixStatus	Long	0=Default - List Name 1=Don't Modify Subject 2=Use Custom Prefix
	SubjectPrefix	String(256)	
(Reserved)	Owner	String(256)	

	HelpMessageFileStatus	Long	Will send a list of commands back to the user when they send help as the password help
(Reserved)	HelpMessageFile	String(256)	
(Reserved)	RemovalMessageFileStatus	Long	
(Reserved)	RemovalMessageFile	String(256)	
	ReplyToMode	Long	0=Replies to List 1=Replies to Sender 2=Replies to Moderator
(Reserved)	MaxMessageSize	Long	
	PostingMode	Long	0=Members can Post 1=Anyone can Post 2=Password Protected Posting
	SubscriptionMode	Long	0=Anyone can Subscribe 1=Subscription is disabled
(Reserved)	AuthenticationMode	Long	
	Password	String(256)	Contains the password if list is password protected- Password is enclosed in [braces] in the subject
(Reserved)	DigestMode	Long	
(Reserved)	DigestMailbox	String(256)	
(Reserved)	DigestAnnotationMode	Long	
(Reserved)	DigestAttachmentMode	Long	
(Reserved)	DigestMessageSeparationMode	Long	
(Reserved)	DigestSchedulingStatus	Long	
(Reserved)	DigestSchedulingMode	Long	
(Reserved)	DigestSchedulingInterval	Long	
(Reserved)	FromAddressMode	Long	

Note: Items marked as reserved may not have been implemented in current releases and are provided for forward compatibility.

Functions

FindFirstList() As Long

FindNextList() As Long

AddList() As Long

GetList() As Long

RemoveList() As Long

EditList(ByVal NewDescription As String, ByVal NewAccountName As String, ByVal NewListName As String, ByVal NewListType As Long, ByVal NewListStatus As Long, ByVal NewHeaderAnnotationStatus As Long, ByVal NewHeaderAnnotation As String, ByVal NewFooterAnnotationStatus As Long, ByVal NewFooterAnnotation As String, ByVal NewModeratorAddress As String, ByVal NewListAddress As String, Optional ByVal NewSubscribeMessageFileStatus As Long = -1, Optional ByVal NewSubscribeMessageFile As String = "(Nil)", Optional ByVal NewUnsubscribeMessageFileStatus As Long = -1, Optional ByVal NewUnsubscribeMessageFile As String = "(Nil)", Optional ByVal NewSubjectSuffixStatus As Long = -1, Optional ByVal NewSubjectSuffix As String = "(Nil)", Optional ByVal NewSubjectPrefixStatus As Long = -1, Optional ByVal NewSubjectPrefix As String = "(Nil)", Optional ByVal NewOwner As String = "(Nil)", Optional ByVal NewHelpMessageFileStatus As Long = -1, Optional ByVal NewHelpMessageFile As String = "(Nil)", Optional ByVal NewRemovalMessageFileStatus As Long = -1, Optional ByVal NewRemovalMessageFile As String =

"(Nil)", Optional ByVal NewReplyToMode As Long = -1, Optional ByVal NewMaxMessageSize As Long = -1, Optional ByVal NewPostingMode As Long = -1, Optional ByVal NewSubSubscriptionMode As Long = -1, Optional ByVal NewAuthenticationMode As Long = -1, Optional ByVal NewPassword As String = "(Nil)", Optional ByVal NewDigestMode As Long = -1, Optional ByVal NewDigestMailbox As String = "(Nil)", Optional ByVal NewDigestAnnotationMode As Long = -1, Optional ByVal NewDigestAttachmentMode As Long = -1, Optional ByVal NewDigestMessageSeparationMode As Long = -1, Optional ByVal NewDigestSchedulingStatus As Long = -1, Optional ByVal NewDigestSchedulingMode As Long = -1, Optional ByVal NewDigestSchedulingInterval As Long = -1, Optional ByVal NewFromAddressMode As Long = -1) As Long

Remarks

Functions return a value of 1 for success and 0 for failure. Other status codes may be returned as information on errors. The header and footer file needs to be located in the annotations subdirectory of the post office configuration directory.

Class

MEAOLS.ListMember

Properties

Wildcard	Name	Type	Description
Yes	Address	String(256)	The address of the member
	AccountName	String(128)	Unused
	ListName	String(128)	Unused
	ListMemberType	Long	Unused
	Status	Long	Unused

Functions

FindFirstListMember() As Long

FindNextListMember() As Long

AddListMember() As Long

GetListMember() As Long

RemoveListMember() As Long

EditListMember(NewAddress, NewAccountName, NewListName, NewListMemberType, NewStatus) As Long

Remarks

Functions return a value of 1 for success and 0 for failure. Other status codes may be returned as information on errors.

3.5 Post Office Administration

Class

MEAOPO.Group

Properties

Wildcard	Name	Type	Description
Yes	RecipientAddress	String(1024)	The address of the group
	Postoffice	String(128)	The account/post office
Yes	GroupName	String(128)	Name of the group
	GroupFile	String(128)	Unused
	GroupStatus	Long	0=Disabled, 1=Enabled

Functions

FindFirstGroup() As Long

FindNextGroup() As Long

AddGroup() As Long

GetGroup() As Long

RemoveGroup() As Long

EditGroup(ByVal NewRecipientAddress As String, ByVal NewPostoffice As String, ByVal NewGroupName As String, ByVal NewGroupFile As String, ByVal NewGroupStatus As Long) As Long

Remarks

Functions return a value of 1 for success and 0 for failure. Other status codes may be returned as information on errors.

Class

MEAOPO.GroupMember

Properties

Wildcard	Name	Type	Description
Yes	Address	String(256)	The address of the member
	Postoffice	String(128)	Unused
	Mailbox	String(128)	Unused

Functions

FindFirstGroupMember() As Long

FindNextGroupMember() As Long

AddGroupMember() As Long

GetGroupMember() As Long

RemoveGroupMember() As Long

EditGroupMember(ByVal NewAddress As String, ByVal NewPostoffice As String, ByVal NewMailbox As String) As Long

Remarks

Functions return a value of 1 for success and 0 for failure. Other status codes may be returned as information on errors.

Class

MEAOPO.Mailbox

Properties

Wildcard	Name	Type	Description
	Postoffice	String(128)	The account/post office
Yes	Mailbox	String(64)	The name of the mailbox
	RedirectAddress	String(512)	The address list to redirect all inbound email to
	RedirectStatus	Long	When the mailbox is redirected. 0=Disabled 1=Enabled 2=Redirect and keep a copy of the message in the mailbox
	Status	Long	0=Disabled, 1=Enabled
	Limit	Long	Size limit (in kilobytes) of the mailbox
	Size	Long	Current size of Inbox in kilobytes

Functions

FindFirstMailbox() As Long

FindNextMailbox() As Long

AddMailbox() As Long

GetMailbox() As Long

RemoveMailbox() As Long

EditMailbox(ByVal NewPostoffice As String, ByVal NewMailbox As String, ByVal NewRedirectAddress As String, ByVal NewRedirectStatus As Long, ByVal NewStatus As Long, ByVal NewLimit As Long, ByVal NewSize As Long) As Long

Remarks

Functions return a value of 1 for success and 0 for failure. Other status codes may be returned as information on errors.

RedirectAddress is a semi-colon delimited list of MailEnable formatted email address, eg:

[\[SMTP:address1@domain.com\];\[SMTP:address2@domain.com\]](#)

Class

MEAOPO.Postoffice

Properties

Wildcard	Name	Type	Description
Yes	Name	String(128)	Name of the postoffice
	Status	Long	0=Disabled, 1=Enabled
Yes	Account	String(64)	Account for the postoffice

Functions

GetMailRootDirectory()

GetConfigurationDirectory() As String

FindFirstPostoffice() As Long

FindNextPostoffice() As Long

AddPostoffice() As Long

GetPostoffice() As Long

RemovePostoffice() As Long

EditPostoffice(ByVal NewName As String, ByVal NewStatus As Long, ByVal NewAccount As String) As Long

Remarks

Functions return a value of 1 for success and 0 for failure. Other status codes may be returned as information on errors. Currently, set the Account to be the same as the postoffice name. You are able to match on the following properties:

Name

Account

3.6 POP Administration

Class

MEAOPS.Access

Properties

Wildcard	Name	Type	Description
	Mode	Long	0=Use Access file, 1=Use Deny file
Yes	AddressMask	String(512)	IP address list to allow/deny
Yes	Status	Long	0=Disabled, 1=Enabled
Yes	Account	String(128)	Account/postoffice
Yes	Right	String(64)	Not used

Functions

AddAccess() As Long

GetAccess() As Long

EditAccess(ByVal NewMode As Long, ByVal NewAddressMask As String, ByVal NewStatus As Long, ByVal NewAccount As String, ByVal NewRight As String) As Long

RemoveAccess() As Long

FindFirstAccess() As Long

FindNextAccess() As Long

Remarks

Functions return a value of 1 for success and 0 for failure. Other status codes may be returned as information on errors. The POP service only has one dedicated class to it, since it leverages the authentication class in order to allow a client to log on. The Access class determines the address of those who are either denied or allowed access to the service. The Mode property determines whether you are going to read the denied address or the allowed addresses file. Be aware that this does not determine whether the file will be used in access or deny mode. The AddressMask is a list of IP addresses that are to be denied/allowed. These can have the * wildcard (so you can use 192.168.0.* for instance). The list is comma delimited.

3.7 POP Retrieval Administration

Class

MEAOPC.POPRetriever

Properties

Wildcard	Name	Type	Description
	LocalPostoffice	String(128)	Local Postoffice to deliver to
	LocalMailbox	String(64)	Local mailbox to deliver to
	MailServer	String(512)	Remote mail server name
	Port	Long	Port of remote server
	Status	Long	0=Disabled, 1=Enabled
	LeaveOnServer	Long	0=Don't delete messages after downloading, 1=Delete messages after downloading
	APOP	Long	0=Don't attempt to use APOP authentication, 1=Use APOP authentication if possible
	UserName	String(128)	Username for remote account
	Password	String(64)	Password for remote account
	DownloadNewOnly	Long	0=Download all messages from remote account, 1=Only download new messages
	CheckEvery	Long	Not used
	LastUsed	Long	Not used
	DownloadedMessagesFile	String(64)	Filename for history file. Leave blank.
	Host	String(64)	Remote MailEnable server if using remote administration

Functions

FindFirstPOPRetriever() As Long

FindNextPOPRetriever () As Long

AddPOPRetriever () As Long

GetPOPRetriever () As Long

RemovePOPRetriever () As Long

EditPOPRetriever (ByVal NewLocalPostoffice As String, ByVal NewLocalMailbox As String, ByVal NewMailServer As String, ByVal NewPort As Long, ByVal NewStatus As Long, ByVal NewLeaveOnServer As Long, ByVal NewAPOP As Long, ByVal NewUserName As String, ByVal NewPassword As String, ByVal NewDownloadNewOnly As Long, ByVal NewCheckEvery As Long, ByVal NewLastUsed As Long) As Long

Remarks

Functions return a value of 1 for success and 0 for failure.

3.8 SMTP Administration

Class

MEAOSM.Access

Properties

Wildcard	Name	Type	Description
	Mode	Long	0=Use Access file, 1=Use Deny file
Yes	AddressMask	String(512)	IP address list to allow/deny
Yes	Status	Long	0=Disabled, 1=Enabled
Yes	Account	String(128)	Account/postoffice
Yes	Right	String(64)	Not used

Functions

AddAccess() As Long

GetAccess() As Long

EditAccess(ByVal NewMode As Long, ByVal NewAddressMask As String, ByVal NewStatus As Long, ByVal NewAccount As String, ByVal NewRight As String) As Long

RemoveAccess() As Long

FindFirstAccess() As Long

FindNextAccess() As Long

Remarks

Functions return a value of 1 for success and 0 for failure. Other status codes may be returned as information on errors.

Class

MEAOSM.Blacklist

Properties

Wildcard	Name	Type	Description
Yes	TargetDomainName	String(512)	Address the email is destined for
Yes	BannedDomainName	String(512)	List of banned addresses
	Status	Long	0=Disabled, 1=Enabled
Yes	Account	String(128)	Account/postoffice

Functions

AddBlacklist() As Long

GetBlacklist() As Long

EditBlacklist(NewTargetDomainName As String, ByVal NewBannedDomainName As String, ByVal NewStatus As Long, ByVal NewAccount As String) As Long

RemoveBlacklist() As Long

FindFirstBlacklist() As Long

FindNextBlacklist() As Long

Remarks

Functions return a value of 1 for success and 0 for failure. Other status codes may be returned as information on errors.

Class

MEAOSM.Domain

Properties

Wildcard	Name	Type	Description
Yes	DomainName	String(512)	Domain name
	Status	Long	0=Disabled, 1=Enabled
	DomainRedirectionStatus	Long	Whether redirection for domain is active 0=Disabled 1=Enabled 2=Redirect from authenticated senders only
	DomainRedirectionHosts	String(2048)	List of hosts to redirect to. This is a comma delimited list of the host addresses.
Yes	AccountName	String(128)	Account/post office

Functions

AddDomain() As Long

GetDomain() As Long

EditDomain(ByVal NewDomainName As String, ByVal NewStatus As Long, ByVal NewDomainRedirectionStatus As Long, ByVal NewDomainRedirectionHosts As String, ByVal NewAccountName As String) As Long

RemoveDomain() As Long

FindFirstDomain() As Long

FindNextDomain() As Long

Remarks

Functions return a value of 1 for success and 0 for failure. Other status codes may be returned as information on errors.

Class
 MEASO.Option

Properties

Wildcard	Name	Type	Description								
	Scope	Long	0=System Wide Value 1=Post Office Value 2=Post Office Mailbox Value								
	Query	String(512)	Query string to denote the URI for the value to be retrieved/set (according to the Scope): <table border="1" data-bbox="981 689 1332 828"> <thead> <tr> <th>Scope</th> <th>Setting</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Category</td> </tr> <tr> <td>1</td> <td>PostofficeName</td> </tr> <tr> <td>2</td> <td>Postoffice/Mailbox</td> </tr> </tbody> </table>	Scope	Setting	0	Category	1	PostofficeName	2	Postoffice/Mailbox
Scope	Setting										
0	Category										
1	PostofficeName										
2	Postoffice/Mailbox										
	ValueName	String(512)	The name of the value or setting to be set/retrieved								
	Value	String(2048)	The value of the setting either to be set or retrieved.								

Functions

SetOption() As Long
 GetOption() As Long

Example:

This example outlines how to enable Web Administration for a Post Office:

```
Dim oMEOption As Object
Set oMEOption = CreateObject("MEASO.Option")
oMEOption.Query = "MailEnable" 'Postoffice Name
oMEOption.Scope = 1
oMEOption.ValueName = "WebAdmin-Enabled"
oMEOption.Value = 1 '1=On, 0=Off
oMEOption.SetOption
Set oMEOption = Nothing
```

Remarks

Functions return a value of 1 for success and 0 for failure. Other status codes may be returned as information on errors.

This API results in settings being stored in the following files/locations:

- Postoffice specific settings are stored in the POSTOFFICE.SYS file under the respective Postoffice branch of the MailEnable\CONFIG directory.

- Mailbox specific settings are stored in the MAILBOXES\{MailboxName}.SYS file under to Postoffice CONFIG directory.
- System settings are stored in the {CategoryName}.SYS file under the CONFIG directory.

The following table contains a list of the value names and their respective meanings for Postoffice specific values:

Postoffice Value Name	Description
MappedDomainEnabled	Determines whether the specified postoffice is mapped to a Security Windows Domain
MappedDomain	Specifies the Windows Security Realm to be used for Integrated Authentication
UPNEnabled	Specifies whether Integrated Authentication for this postoffice supports UPN (User@securitydomain) formatted usernames
WindowsAuthenticationEnabled	Specifies whether the postoffice is mapped to a Windows Security Realm for Integrated Authentication
WindowsAccountAutoCreation	Specifies whether accounts are to be automatically created if the user has authenticated using Integrated Authentication
WebAdmin-Enabled	Specifies whether Web Administration is available for ADMIN or SYSADMIN users of this Post Office.
WebAdmin-CanEditMailboxes	Determines whether ADMIN or SYSADMIN users are able to edit details of Mailboxes
WebAdmin-MaxMailboxes	Determines the maximum number of Mailboxes that can be added via Web Administration for this Post Office.
WebAdmin-DefaultMailboxSize	Determines the maximum Mailbox Quota Size that can be specified when creating or modifying a Mailbox
WebAdmin-CanEditMailboxSize	Determines whether ADMIN or SYSADMIN users are able to specify Mailbox sizes
WebAdmin-CanEditLists	Determines whether ADMIN or SYSADMIN users are able to edit the details for lists
WebAdmin-MaxLists	Determines the maximum number of lists that can be managed by an ADMIN or SYSADMIN user
WebAdmin-MaxListMembers	Determines the maximum number of list members that can be assigned to a list
WebAdmin-CanEditDomains	Specifies whether ADMIN or SYSADMIN users are able to configure domain details via Web Administration

The following table contains a list of the value names and their respective meanings for mailbox specific values:

Postoffice Value Name	Description
CharSet	Default charset for mailbox
ReplyAddress	The default SMTP reply address
DisplayName	The friendly name/display name for the mailbox
MailBox-DropEventStatus	0=Mailbox delivery event disabled 1=Mailbox delivery event enabled
SMTP	0=SMTP disabled for mailbox 1=SMTP enabled for mailbox
POP	0=POP disabled for mailbox 1=POP enabled for mailbox
HTTPMail	0=HTTPMail disabled for mailbox

	1=HTTPMail enabled for mailbox
WebMail	0=WebMail disabled for mailbox 1=WebMail enabled for mailbox
IMAP	0=IMAP disabled for mailbox 1=IMAP enabled for mailbox
TimeZone	Timezone for mailbox
AutoSignatureStatus	0=Autosignature disabled 1=Autosignature enabled
DefaultAddress	Default SMTP address for the mailbox
MsgFormat	TEXT=Use text editing for webmail HTML=Use HTML editing for webmail
WebMail-MessagesPerPage	Number of messages per page in webmail
WebMail-UseDeletedItemsFolder	Deleting messages moves them to deleted items folder in webmail
WebMail-ClearDelectedOnLogOff	Deletes the contents of the deleted items folder in webmail when the user logs off
WebMail-AllowNewWindows	0=Open messages within inbox frame 1=Open messages in a new window
SMTP-Inbound-Message-UsageRestrictionEnabled	0=User has no throughput restrictions 1=User has message throughput restrictions
SMTP-Inbounce-Message-UsageRestriction	The number of messages per hour the mailbox can send
MailBox-MailboxRulesSatus	0=Mailbox filters disabled 1=Mailbox filters enabled

3.9 Remote Administration

3.9.1 Registering the Host

Before you can manage a server, its credentials must be added to the machine that is requesting administration. To do this you need to use the `MEAORA.Hosts` class to add the host and generate its credentials.

A sample for registering a host follows:

```
Dim oMEAORAHost As Object
Set oMEAORAHost = CreateObject("MEAORA.Hosts")
oMEAORAHost.HostName = Me.lstIPAddress
oMEAORAHost.Address = Me.lstIPAddress
oMEAORAHost.UserName = Me.txtUsername
oMEAORAHost.Password = Me.txtPassword
oMEAORAHost.Port = Me.txtPort
oMEAORAHost.AddHost
Set oMEAORAHost = Nothing
```

3.9.2 Registering a Session for Remote Administration

MailEnable Enterprise allows you to cluster hosts to share the same configuration repository. This is done using the `MEAORA.Session` class. This will query the host you are connected to using the credentials passed using the `MEAORA.Hosts` object. It uses the `Address` property of both objects as a common key. Hence, the procedure is to call `MEAORA.Hosts::AddHost` first, then `MEAORA.Session::Authenticate`.

Not only does this allow you to authenticate against the remote host, but it also caches the credentials for any of the hosts defined within the same cluster as the host you are connecting to.

```
Dim oMEAORASession As Object
Set oMEAORASession = CreateObject("MEAORA.Session")
oMEAORASession.Address = strIPAddress
Select Case oMEAORASession.Authenticate
Case 0:
    MsgBox "Communications Failure"
Case 1:
    MsgBox "Service authenticated user successfully."
Case 2:
    MsgBox "Remote Host does not appear to be running MailEnable
Remote Administration Service"
Case 3:
    MsgBox "Service was contacted but account used was invalid.
Check the password for the account and ensure that the user has SYSADMIN"
```

```
rights."  
    End Select  
    Set oMEAORASession = Nothing
```

3.9.3 Calling the Remote Object

Now that you have authenticated the remote session, you need to modify your existing instantiations of the MEAOXX objects to set the host to which it should connect to.

An example follows:

```
Dim oMEAOAM As Object  
Set oMEAOAM = CreateObject("MEAOAM.AddressMap")  
oMEAOAM.CurrentHost = "127.0.0.1"  
oMEAOAM.SetHost  
'  
' Your code goes here  
'  
Set oMEAOAM = Nothing
```

The object will retain the host setting after each call is made, hence you will need to change it whenever you want to manage a different host. To manage the local machine, you should set the CurrentHost property to blank. This will ensure that the local configuration providers are used rather than passing calls through the remote management service.

3.9.4 Accessing Miscellaneous Configuration Settings

Remote Management introduces some additional configuration objects that are not defined in the existing API. These specifically deal with message storage, system services and system settings (registry settings). These are defined in this section.

3.10 Server Management Groups and Clusters

MailEnable Enterprise Edition allows you to define a logical Management Group or Cluster of MailEnable servers. In MailEnable Professional Edition, the only server name that appeared under the Servers node in the administration program was localhost. MailEnable Enterprise Edition allows you to register additional hosts under the Servers branch in the administration program.

The following code allows you to list all the servers that are currently defined in the same server group as the server we are querying (in this case the server at 127.0.0.1).

```
Dim oMEAOSVServer
Set oMEAOSVServer = CreateObject("MEAOSV.Server")
oMEAOSVServer.Host = "127.0.0.1" ' Host you want to connect to
oMEAOSVServer.SetHost 'Tell the provider you want to set to this
host
oMEAOSVServer.Status = -1
oMEAOSVServer.Port = -1
oMEAOSVServer.DisplayName = ""
oMEAOSVServer.Address = ""
If oMEAOSVServer.FindFirstServer = 1 Then
Do
    '
    ' Need to ensure that we load the credentials for the
servers
    '
    Me.lstClusterNodes.AddItem oMEAOSVServer.Address
    oMEAOSVServer.Status = -1
    oMEAOSVServer.Port = -1
    oMEAOSVServer.DisplayName = ""
    oMEAOSVServer.Address = ""
Loop While oMEAOSVServer.FindNextServer = 1
End If
Set oMEAOSVServer = Nothing
```

Note: Like all MailEnable Objects, you can use the SetHost property to determine which server you are querying.

4 Developing Connectors

MailEnable allows you to define custom connectors to facilitate mail connectivity to backend systems. Such connectors are useful to integrate the likes of telephony/communications, printing and external systems with MailEnable.

For example, it may be desirable to publish access to a backend order entry system as an SMTP address to allow orders to be distributed to a backend system. Orders would be received by SMTP and placed in a backend Order Management System. Backend system could also place messages into its own inbound queue for distribution/notification, etc.

4.1 Implementing a Connector

To implement your own connector, you must first give the connector a name.

In our example, we will use the MEOES as the connector name (for Order Entry System).

STEP 1: Creating the Connector Queues

Firstly, you must create the following Queue directories:

```
C:\Program Files\Mail Enable\Queues\MEOES\Inbound\Messages  
C:\Program Files\Mail Enable\Queues\MEOES\Outbound\Messages
```

Note: These assume the default location for the Queue directory. It is also important to make sure that the entire paths outlined above exist. Specifically, the Messages Subdirectories are imperative.

STEP 2: Registering the Connector

You then need to create the following registry key:

```
HKEY LOCAL MACHINE\SOFTWARE\Mail Enable\Mail Enable\Connectors\MEOES
```

STEP 3: Allocating Mailboxes Connector Addresses

Now that the connector queues are created you are able to add address maps to that connector in the ADDRESS-MAP.TAB file. This is exactly how the SMTP, POPC, PostOffice and List Connectors work.

An Example address map follows:

Message arrives via SMTP -> MTA determines connector mapping -> MEOES connector
-> System

This is represented in the address map file as:

Source Address = "[SMTP:OrderSystemPayment@domain.com]"

Target Address = "[MEOES:PAYMENT]"

You can manage address maps using the AddressMap Object (outlined in the previous section).

This rule would mean that any messages arriving for the SMTP Address would be placed in the Queue with a command file addressing to [MEOES:PAYMENT], with a corresponding message in the Messages directory. The reverse can also be achieved.

STEP 4: Creating the Connector

You now need to write the connector itself. This is simply a program that reads to and from the MEOES queues you just created.

A good starting point for writing your own connector is the sample MTA pickup event provided on the MailEnable Web Site. The MailEnable System Manual also explains how connectors work.

4.2 Considerations

When developing custom connectors you should try to avoid using generic connector names - ie: connector names that are logically intended to be part of MailEnable. These would specifically include FAX, PRN, SMS and TEL queue names. Any custom queue names should be prefixed with at least a two vendor allocated letters to distinguish it from other queues.

This will prevent any custom connector conflicting with address maps of any integrated connectors, as well as reducing the likelihood of cross vendor conflicts.

5 Developing MTA Pickup Events

When the MTA moves a message between connectors, an optional executable file or COM DLL can be invoked. This is called a pickup event. The MTA pickup event will pass the mail message filename to the external application/COM DLL.

For example, if you wrote a VB script that adds some text to the end of each email you could activate this application through the pickup event. The application/DLL receives the messagefilename and connectortype as parameters.

eg: Program MessageFileName ConnectorType

Where:

Program is the program filename,
MessageFileName is the name of the message file
ConnectorType is the type of messages (ie. SMTP, LS, SF).

Be aware that the directory path to the message is not passed to the program. You will need to read the directory path from the registry in the external application.

5.1 Developing a Pickup Event Application

A Pickup Event Application is a Windows Executable that is shelled as a message passes through the MTA. As mentioned earlier, the shelled executable accepts a space delimited set of parameters.

Since the parameters do not contain the physical queue path, we first need to access the registry and determine the location of the connectors queue and construct the location of the message command file and the message itself.

Simple Visual Basic Example:

```
Sub Main()  
.....  
' This routine is unsupported and is provide for reference purposes only  
,  
' This primitive example checks mta messages for .exe in the message  
' contents and deletes any files that do!  
,  
.....  
Dim sMsgCommandFile as String  
Dim sMsgFile as String  
Dim hFile As Long  
Dim args() As String  
Dim sFileLine as String  
  
args() = Split(Command(), " ")
```

```
sMsgCommandFile = GetRegistryString("SOFTWARE\Mail Enable\Mail Enable", "Data Directory") & _
"\QUEUES\" & args(1) & "\Inbound\" & args(0)
sMsgFile = GetRegistryString("SOFTWARE\Mail Enable\Mail Enable", "Data Directory") & _
"\QUEUES\" & args(1) & "\Inbound\Messages\" & args(0)
hFile = FreeFile
bPerformAction = False
On Error goto Err Handler
Open sMsgFile For Input as #hFile
While Not EOF(hFile)
Line Input #hFile, sFileLine
if Instr(1, ICase(sFileLine), ".exe") Then
bPerformAction = True
Exit While
end if
Wend
Close (hFile)
if PerformAction = True Then
Kill(MsgCommandFile)
Kill(MsgFile)
Exit Sub

ErrorHandler:
App.LogEvent "Could not process pickup event for Connector: " & ConnectorCode & _
" Message ID: " & MessageID
End Sub
```

Note: There are other examples of MTA Pickup Events available at <http://www.mailenable.com/developers> or under the third party utilities section of the web site.

5.2 Developing a Pickup Event Class

MailEnable's MTA also allows you to call COM components as well as shelled MTA pickup events. The advantage of calling a COM component is that it can run in the same context and process address space as the MTA process itself.

Using the COM Pickup Event also can have considerable disadvantages (but careful programming of the Pickup Event can overcome the issue).

A considerable disadvantage in calling the COM DLL is that any faults occurring in the COM object will be inherited by the MTA and therefore could make the MTA unstable. To avoid this, authors of such COM components should take care to manage exceptions/errors and ensuring that they do not cause an unknown state.

5.2.1 Developing the Object

The most practical/simplest tool for developing a COM Pickup Event is Visual Basic 6.0 You should create a new project as an ActiveX DLL and name the project/object and classes to be indicative to the purpose of the pickup event.

You should then add a function to one of the classes as a public Method called "Execute". The function should take a string as a parameter.

Example:

```
Public Function Execute (ByVal Params As String) As Long
    '
    'Params: Same parameters as those passed to a pickup event.
    '
    '
    ' Function should return 1 if Success, 0 if failure
    '
End Function
```

When a message passes through the MTA, the pickup event will call the execute method on the COM object, passing the same parameters as are passed to a shelled Pickup Event.

The simplest way to test the functionality is to run the MTA in debug mode on your development server and have the Execute function simply display a message box.

Eg:

```
Public Function Execute (ByVal Params As String) As Long
    '
    'Params: Same parameters as those passed to a pickup event.
    '
    MsgBox Params
    '
    ' Function should return 1 if Success, 0 if failure
    '
    Execute = 1
End Function
```

When developing the pickup event for production, it is important that the respective and Microsoft recommended practices for creating unattended library COM components are followed. Specifically, the component should not display and UI and should be compiled with the Unattended Execution option.

5.2.2 Integrating the COM Pickup Event

To configure the MTA to shell a COM Component, you need to enter the qualified object.class name into the following registry key

KeyRoot	HKLM\SOFTWARE\Mail Enable\Mail
---------	--------------------------------

	Enable\AGENTS\MTA
Value Name	Pickup Event Class Name
Value	String containing class name
Value Type	REG_SZ
Example Value	MyObject.MTAPE

You also need to configure the MTA to call the COM Pickup Event via an additional Registry Key:

KeyRoot	HKLM\SOFTWARE\Mail Enable\Mail Enable\AGENTS\MTA
Value Name	Pickup Event Class Enabled
Value	0 = Off, 1 = On
Value Type	REG_DWORD
Example Value	MyObject.MTAPE

6 Developing Mailbox Delivery Events

A Mailbox Delivery Event is optionally triggered when the Postoffice Connector attempts to deliver mail to a mailbox. The Postoffice Connector will check whether a delivery event is configured for the mailbox and will execute the specified executable.

When the delivery event is triggered, the following parameters are passed on the command line to the specified executable (they are delimited by a space):

PostofficeName Mailbox MessageID

Note: You should note that Mailbox Delivery Events generate different parameters to Mail Transfer Agent Pickup Events. You cannot therefore simply run code designed for a Pickup Event as a Delivery Event (although it should be quite simple to modify the parameter list expected by the target executable).

7 SendMail COM Component

The COM component allows easy integration of emailing sending from within any COM supporting application. It not only supports sending email to a MailEnable server, but also can be used to send email to any SMTP compatible mail server.

Properties

Property	Explanation
AttachmentFilename	The name of the file that you wish to add as an attachment.
AttachmentName	The name you wish to call the attachment.
ContentType	The ContentType of the email you are trying to send. For instance, if you wish to send a HTML email, use this property to set the content type to HTML.
ErrorString	This contains the full English language description of the last error. If you encounter an error, you can check this string for a more detailed error.
MailBCC	This is list of email addresses to BCC the email to. When using multiple email addresses, separate them with a semi-colon ";".
MailBCCDisplayName	This is list of email addresses that are the display name corresponding to the email address you have set in MailBCC. This list is optional. When using multiple email addresses, separate them with a semi-colon ";".
MailCC	This is list of email addresses to CC the email to. When using multiple email addresses, separate them with a semi-colon ";".
MailCCDisplayName	This is list of email addresses that are the display name corresponding to the email address you have set in MailCC. This list is optional. When using multiple email addresses, separate them with a semi-colon ";".
MailFrom	This is the email address of the person you want as the sender.
MailFromDisplayName	The display name of the from MailFrom email address.
MailTo	The email address to send the email to. If you wish to send to multiple email addresses, separate the emails with a semi-colon ";".
MailToDisplayName	This is the display name that will be shown as the To address. It is usually the full name of the person you are sending to (i.e. "John Smith")
Messagebody	The message contents.
MessageBodyText	An optional property used to force the content for the textual content of the message. If the property is not set, MailEnable will generate a textual version of the message from the HTML content supplied (assuming the ContentType is set as text/html).
Server	The email server to connect to. If none is supplied it will try to connect to the local machine.
ServerPort	The port to connect to. The default is 25.
Subject	The subject of the email message.

Methods

Method	Explanation
AddHeader	Adds a custom header to the email. Be careful when using this

	function, as incorrectly formed headers could prevent the mail from being sent. The Reply-To header is an example of a header that can be set via this method.
ClearHeaders	Clears any custom headers that have been added with AddHeader. This would be used if you were sending more than one message (you put this call between your sends).
SendMessage	Send the email that has been configured with the options. The function will return zero for failure and number greater than zero for success.
SetDefault	This will clear all previous settings to default. If sending multiple emails, use this between the calls.
AddAttachment (Filename, AttachmentName)	This will add an attachment to the message being composed. The first parameter is the Path to the file that you want to attach to the message (eg: C:\My Documents\test.txt). The second parameter is the name that the attachment should have when read via a mail client (eg: test.txt)
ClearAttachments	This will remove all attachments from the message being composed.

If you are attempting to send mail from an external application or web page (including using the COM component (MEASP) or the Command Line Send Utility) and want to send mail to a remote user or mail server, you will need to configure the relay permissions of the server to allow the application to relay.

For example: To allow the local machine (and the COM object) to send out email, you need to add the 127.0.0.1 IP address as a privileged IP address.

Please follow these instructions:

1. Load the Administration program, expand the **Servers\Localhost\Connectors** branch.
2. Right click on the SMTP icon, select **Properties** from the popup menu, then click the **Relay** tab.
3. Enable Allow relay for privileged IP ranges.
4. Click **Privileged IPs** and in the window that appears, make sure that you have selected **By Default all computers will be Denied relay rights**, and add the source IP address that your component will use when it attempts to send mail through MailEnable's SMTP Connector.

You need to check your MailEnable SMTP Debug and Activity logs to verify the IP address your component/application is actually attempting to connector on port. Also, these logs should contain errors that will and allow you to debug. You need to ensure that the IP Address that the program/agent is using to communicate with MailEnable is granted relay rights.

Examples

Sending a HTML email from an ASP page:

```
<%
```

```
Dim oMail
Set oMail = server.CreateObject("MEMail.Message")
oMail.MailFrom = "peter@mailenable.com"
oMail.MailFromDisplayName = "Test Account"
oMail.ContentType = "text/html;"
oMail.MailTo = "peter@mailenable.com"
oMail.Subject = "Welcome to our service"
oMail.AddHeader("Reply-To: <youraddress@yourdomain.com>")
oMail.MessageBody = "<html><body><h1>Hello there,<BR>Welcome to our
new service.</h1></body></html>"
oMail.SendMessage
%>
```

Sending an email with an attachment:

```
<%
Dim oMail

set oMail = server.CreateObject("MEMail.Message")

oMail.MailFrom = "peter@mailenable.com"
oMail.MailFromDisplayName = "Update Account"
oMail.MailTo = "customer@mailenable.com"
oMail.Attachmentfilename = "c:\documents\updateinfo_14_4.zip"
oMail.Attachmentname = "updateinfo.zip"
oMail.Subject = "New update information"
oMail.MessageBody="Find the new info attached."
oMail.SendMessage
%>
```

By setting the *ContentType* value to text/html, the component will generate a HTML and Plan Text representation of your message encapsulated in MIME format.

You need only to set the *ContentType* property to text/html and, when the *SendMessage* method is called, the component generates the MIME encapsulated message with a multipart alternative content boundary. This boundary then contains respective text/plain and text/html boundaries.

The mail client then determines which of the alternative content types it wants to read - based on the capabilities of the mail client or the users settings.

If you set the *MessageBody* and *MessageBodyPlain* properties of the component, it will not generate a textual representation of the message and will use the property value specified for *MessageBodyPlain*.

8 Developing Skins

MailEnable Professional and Enterprise contains WebMail and WebAdmin applications that are driven by Microsofts Internet Information Services and .NET.

MailEnable Web Applications have two core components, Skins and Bases. These are outlined as follows:

Skins: Skins are a collection of graphics and styles that change the way a web application looks. You can heavily modify the graphics, fonts and colour scheme of your web application by creating your own skin.

Bases: Bases are the ASPX pages themselves that make up the web application. These files determine the layout and functionality application. You should not modify the ASPX files distributed with MailEnable as the files will be overwritten by upgrading or reinstalling the software.

Each base can have multiple skins. Any skin created correctly for the default base is compatible with web administration.

8.1 Developing Skins

As mentioned earlier, a Skin is a collection of images and styles that represents the branding and graphical treatment of the web application. Skins are comprised of an array of named images and a CSS stylesheet (which stores the colour scheme, font usage, etc).

Skins are tied to a particular base application (otherwise known as a layout). MailEnable currently provides a base application known as “hoodoo”. Versions 1 and 2 of MailEnable used two different bases called “default” and “enterprise” but these are no longer supported.

8.1.1 Hoodoo Skins

Images

The images used by MailEnable’s default application are outlined in the following table. Not all dimensions are fixed, which allows you to vary the size. For instance the top menu icons can be varying widths. Check the table below for items that should not be altered in size.

File Name	Dimensions	Description
Add.gif	16x16	Add a new appointment
Addcontact.gif	16x16	Add a contact
Arrowdown.gif	14x10	Indicates reverse sort order for a column in the folder list
Arrowup.gif	14x10	Indicates forward sort order for a column in the folder list
Calendar.gif	16x16	Calendar

Calendar_shared.gif	16x16	
cancel.gif	16x16	Cancel email
Close.gif	16x16	
Clsdfold.gif	16x16	
Clsfold_shared.gif		
compose.gif	16x16	Compose
contact.gif		
corp_logo.gif	182x24	Top left logo when logged in
delete.gif	16x16	Delete selected items
DeletedItems.gif	16x16	Deleted items folder icon for folder list
Download.gif		
Drafts.gif		
Flagged.gif		
folders.gif	16x16	Folders
form.gif	16x16	View headers
Group.gif		
Help.gif		
Inbox.gif	16x16	Closed folder icon for folder list
login-window_logo.gif	362x83	Login graphic
logoff.gif	16x16	Log off
openedletter.gif	16x16	Drafts folder icon for folder list and read letter
options.gif	16x16	Options
paperclip.gif	16x16	Attachment icon
Print.gif	16x16	Print
Reply.gif	16x16	
Reply_all.gif	16x16	
Save.gif	16x16	Save email
send.gif	16x16	Send email
tv_dots.gif	16x20	Treeview dotted join
tv_dotsl.gif	16x20	Treeview vertical join
tv_minusdots.gif	16x20	Treeview expanded join
tv_plusdotsb.gif	16x20	Treeview collapsed join
Upload.gif	16x16	
World.gif	16x16	

Stylesheet

MailEnable also use a stylesheet to manage the colour scheme, font scheme and to fine-tune layout.

Class	Description
#imgCompose, #imgFolders, #imgShares	
#tblMsgOptions.ME_Header	
.AccountUsageLabel	
.clsdivPanelMessageList	
.clsdivPreviewPane	Border around the preview pane
.CommandMenu	Border around the command toolbar
.divActions	
.divScrollable	The rectangle around the folder treeview
.HeaderClass	The header of the message or contact list
.imgCommandMenuBgImages	
.ItemDetailsPanel	
.ME_AlertDialogBoxBorder	

.ME_Body	
.ME_BodyCommandPanel	
.ME_BodyContentPanel	
.ME_BodyHeader	
.ME_BodyHeaderPanel	
.ME_BodyPanel	
.ME_Button	Buttons
.ME_ButtonBG	
.ME_CalendarItem	
.ME_CmdMenuPanel	
.ME_CommandPanel	
.ME_CommandPanelLabel	
.ME_CommandPanelLabel:hover	
.ME_CommandPanelLabel:link	
.ME_CommandPanelLabel:visited	
.ME_ComposeToolBar	
.ME_ConfirmDialogBoxBorder	
.ME_ContentAreaHeaderLeftLabel	Text style for the header over the treeview
.ME_ContentAreaHeaderLeftPad	
.ME_ContentAreaHeaderLeftPadImage	
.ME_ContentAreaHeaderRightLabel	Text style for the header over the message
.ME_ContentAreaHeaderRightPad	
.ME_ContentAreaLeftHeader	The rectangle around the top of the file list. ie. where "Folders" is
.ME_ContentAreaRightHeader	Border around right header
.ME_ContentFrame	
.ME_ContentFrame A	
.ME_ContentPanelContainer	Border around the account summary
.ME_CTable	
.ME_CTable td	
.ME_DatePicker	
.ME_DialogBody	
.ME_DialogBodyText	
.ME_DialogBox	
.ME_DialogTitle	Title of the dialog
.ME_DropDownList	
.ME_DTable	
.ME_DTable td	
.ME_DTable th	
.ME_EmptyList	
.ME_Field	
.ME_FieldLabel	
.ME_FolderPanel	
.ME_FolderPanel A	
.ME_FrameBody	
.ME_FreeTextBox	
.ME_FreeTextBox_OptionsEditor	

.ME_Header	Border around the message list header
.ME_HeaderText	
.ME_Heading	
.ME_HResize	
.ME_HResizeBar	
.ME_HResizer	
.ME_Input	
.ME_InputDialogBoxBorder	
.ME_List0	
.ME_List1	
.ME_ListHdrImage	
.ME_ListHeader	
.ME_ListImage	
.ME_ListRowContainer	
.ME_LoginBody	
.ME_LoginButton	
.ME_LoginButtonHover	
.ME_LoginDialogCanvas	
.ME_LoginDropDownList	
.ME_LoginFrameBody	
.ME_LoginFrameContent	
.ME_LoginFrameContentFooter	
.ME_LoginFrameContentFooterBody	
.ME_MenuCap	Middle part of the tasks/cals/etc
.ME_MenuCapText	
.ME_MenuEndCellLeft	Left part of the tasks/cals/etc
.ME_MenuEndCellRight	Right part of the tasks/cals/etc rectangle
.ME_Message	
.ME_MessageBodyBlock	
.ME_MessageHeader	
.ME_MessageHeader A	
.ME_MessageHeaderBlock	The header details of a message
.ME_MessageHeaderFields	The test of the message header
.ME_MessageHeaderFieldsValues	
.ME_MessageList0	
.ME_MessageList1	
.ME_MessageListContainer	
.ME_MessageListHover	
.ME_MessagesListFooter	Footer underneath the message list
.ME_MessagesListFooterLabel	
.ME_MessagesListHeader	Header of the message list
.ME_MessagesListHeader A	
.ME_MessagesListHeader A:hover	

.ME_MessagesListHeader A:link	
.ME_MessagesListHeader A:visited	
.ME_NoLink	
.ME_Normal	
.ME_Notify	
.ME_NotifyTable_bottom	
.ME_NotifyTable_bottom_left IMG	
.ME_NotifyTable_bottom_right IMG	
.ME_NotifyTable_left	
.ME_NotifyTable_right	
.ME_NotifyTable_top	
.ME_NotifyTable_top_left IMG	
.ME_NotifyTable_top_right IMG	
.ME_NotifyText	
.ME_OptionHeader	
.ME_OptionTable	
.ME_PreviewFrameBody	
.ME_ProgressIMG	
.ME_RecurrenceContainerCell	
.ME_RecurrenceTable	
.ME_ResponseStatus	
.ME_RowSelect	
.ME_Screen	
.ME_SS_ControlLinks	Links for the control menu for the slideshow
.ME_SS_ControlLinks:visited	Links for the control menu for the slideshow
.ME_SS_ImageTitle	Image filenames displayed in the slideshow
.ME_StandardText	
.ME_StandardTextSmall	
.ME_SwapButton	
.ME_SwapButton: hover	
.ME_Tab_body	
.ME_Tab_BodyDiv	
.ME_Tab_button	
.ME_Tab_filler	
.ME_Tab_selected	
.ME_Tab_table	
.ME_Tab_table th	
.ME_Table	
.ME_TableHeading	
.ME_TabOff	
.ME_TabOn	
.ME_TDFBMsg	
.ME_TDFBMsgTxt	

.ME_TextField	
.ME_Toolbar, .ME_Toolbar_nohover	
.ME_Toolbar:hover	
.ME_Toolbar_nohover:hover	
.ME_ToolBarButton	
.ME_UnreadCount	
.ME_UsageBarBackground	Background of the usage bar
.ME_UsageBarFull	Usage bar when full
.ME_UsageBarNormal	Usage bar
.ME_UsageBarPanel	
.ME_UsageLabel	
.ME_VResizeBar	
.ME_VResizer	
.MEActionsButton	
.MEMessageList	
.MEMessageList td	
.MEPopUpPanel	
.MEPopUpTable	
.mescroll	
.MsgCmdBarIMG	
.MsgCmdBarMain	
.MsgCmdBarMainItem_Left	
.MsgCmdBarMainItem_Right	
.MsgList_SelectedRow	
.MsgSummaryCmdBarIMG	
.MsgSummaryCmdBarIMG:hover	
.OptionsPageBody	
.PanelMessageList	Border around the message list
.PopUpRow	
.PopUpRowHoverOver	
.SearchPanel	The search panel
.SummaryDate	
.SummarySubject	
.SummaryTable	
.SwapButtonsContainer	Rectangle around the buttons for Tasks/Cals/etc
.tv_nodeexpand	
.tv_nodelabel	
.tv_selectedrow	
.tv_unselectedrow	
A.ME_AddContactLink	
A.ME_AddContactLink:active	
A.ME_AddContactLink:link	
A.ME_AddContactLink:visited	

A.ME_ColHeader	
A.ME_ColHeader:hover	
A.ME_ColHeader:link	
A.ME_ColHeader:visited	
A.ME_FolderPanel:visited	
A.ME_HeaderText	
A.ME_HeaderText:active	
A.ME_HeaderText:link	
A.ME_HeaderText:visited	
A.ME_SwapButton	
A.ME_SwapButton:active	
A.ME_SwapButton:hover	
A.ME_SwapButton:link	
A.ME_SwapButton:visited	
A:hover	
A:link	
BODY.ME_Canvas	
body.ME_HTMLEditor	
BODY.ME_LoadCanvas	
BODY.ME_Messages	
BODY.ME_Topmenu	
div.ME_PercentCompleteBar	
div.ME_PercentCompleteBorder	
div.suggestions	
div.suggestions div	
div.suggestions div.current	
INPUT	
ME_AddContactLink	
SELECT	
span.suggestionForText	
tblLeftPanel	
TEXTAREA	

8.2 Skins – Additional Considerations

MailEnable Skins are comprised of graphics and one or more cascading stylesheets.

If you add your own items to the stylesheet, remember to always name them. Otherwise, HTML emails that are displayed may be affected. You won't be able to add items to your stylesheet if you are distributing it for general use, as other users will not have any changes you make to any of the web pages. When naming a style that will be applied to a tag, use the prefix ME_ to avoid possible generic name uses. The name of the stylesheet file is me.css.

8.3 Creating a Skin

To create a new skin for a base application, you should create a new directory under the ...\\BIN\\NetWebMail\\hoodoo\\skins directory.

The easiest way to create a new skin is to copy the files associated with the default skin (which is distributed with MailEnable).

This is outlined below:

1. Open the Mail Enable\\bin\\netwebmail\\hoodoo\\skins directory
2. Make a copy of a skin directory. This will become your skin directory.

The name of the directory is the name of the skin. This name is displayed to the user when they have an option for choosing a skin.

8.4 Distributing Skins

In order to distribute your skin to others, you just need to distribute the skin directory you have created. Other users just then need to copy the directory you have created into their skin directory and it is available for use.

8.5 Developing Base Applications

You must be familiar with ASP.NET (or your chosen scripting language) to create new base web applications.

If you wish to develop your own base application (or customize the default base that is distributed with MailEnable), you should create another directory under ..\\Mail Enable\\BIN\\NetWebMail\\ to contain the ASP pages of your modified application

8.6 Licensing Considerations

The license agreement for MailEnable stops you from distributing any MailEnable intellectual property. This means that you cannot redistribute any part of the web mail apart from the skin you have created. This does not stop you from modifying the web pages to your own needs.

You are able to modify the interface for web mail by generating your own skin. You are also allowed to replace any copyright notices displayed in the web mail and/or webadmin applications so long as you use these pages for your own purposes and take reasonable efforts to protect the copyright of the application.

9 Examples

9.1 Creating a Post Office

Example of Creating a Post Office from ASP Script:

```
<%  
Function CreatePostoffice(sPostoffice, sPassword)  
    Dim oWebAdmin  
    Dim oMailbox  
    Dim oLogin  
    Dim IResult As Long  
    CreatePostoffice = False  
    If Len(sPostoffice) > 0 And Len(sPassword) > 0 Then  
        Set oWebAdmin = Server.CreateObject("MEWebAdmin.Postoffice")  
        oWebAdmin.Account = sPostoffice  
        oWebAdmin.Name = sPostoffice  
        oWebAdmin.Status = 1  
        IResult = oWebAdmin.AddPostoffice  
        If (IResult = 1) Then  
            Set oMailbox = Server.CreateObject("MEWebAdmin.Mailbox")  
            oMailbox.Postoffice = sPostoffice  
            oMailbox.Limit = -1  
            oMailbox.Mailbox = "Postmaster"  
            oMailbox.RedirectAddress = ""  
            oMailbox.RedirectStatus = 0  
            oMailbox.Status = 1  
            IResult = oMailbox.AddMailbox  
            If (IResult = 1) Then  
                Set oLogin = Server.CreateObject("MEWebAdmin.Login")  
                oLogin.Account = sPostoffice  
                oLogin.Description = "Postmaster Mailbox"  
                oLogin.Password = sPassword  
                oLogin.Rights = "ADMIN"  
                oLogin.Status = 1  
                oLogin.UserName = "Postmaster@" & sPostoffice  
                IResult = oLogin.AddLogin  
                If (IResult = 1) Then  
                    CreatePostoffice = True  
                End If  
            End If  
        End If  
    End If  
    Set oPostoffice = Nothing  
    Set oMailbox = Nothing  
    Set oLogin = Nothing  
End Function  
  
If CreatePostoffice(Request("PO"), Request("PWD")) Then  
    Response.Write "Created Postoffice"  
Else  
    Response.Write "Failed to Create Postoffice"  
End If  
%>
```

9.2 Deleting a Post Office

Example of Deleting a Post Office using VBScript:

```
Public Function DeletePostoffice(sPostoffice As String)
    'deletes a postoffice and all the directories + content
    On Error GoTo errHandler
    Dim oPostoffice As Object
    Dim oFSO As Object    'New FileSystemObject
    Dim IResult As Long
    Set oPostoffice = CreateObject("MEOAPO.Postoffice")
    Set oFSO = CreateObject("Scripting.FileSystemObject")
    'first we delete all the entries in the files
    'first the postoffice file
    oPostoffice.Account = sPostoffice
    oPostoffice.Name = sPostoffice
    oPostoffice.Status = -1
    IResult = oPostoffice.RemovePostoffice()
    'remove address maps
    DeleteMapping "", sPostoffice
    'remove auth
    RemoveAuth sPostoffice, ""
    'remove domains
    RemoveDomain sPostoffice, ""
    'then we delete all the files
    'delete the postoffice directory in the config directory
    If oFSO.FolderExists(GetMailConfigDirectory() & "\postoffices\" & sPostoffice) Then
        oFSO.DeleteFolder GetMailConfigDirectory() & "\postoffices\" & sPostoffice, True
    End If
    'then the mail directories
    If oFSO.FolderExists(GetMailRoot() & "\" & sPostoffice) Then
        oFSO.DeleteFolder GetMailRoot() & "\" & sPostoffice, True
    End If
    Exit Function
errHandler:
    MsgBox "DeletePostoffice:Error:" & Err.Description
End Function

Public Function DeleteMapping(sDestination As String, sPostoffice As String)
    'remove an address map
    Dim oAddressMap As Object
    Dim IResult As Long
    Set oAddressMap = CreateObject("MEOAAM.AddressMap")
    'first we get the old one
    With oAddressMap
        .Account = sPostoffice
        .DestinationAddress = sDestination
        .Scope = ""
        .SourceAddress = ""
        IResult = .RemoveAddressMap()
    End With
    'Iresult will be 0 if it doesn't delete anything, which may happen if no destination specified
    If IResult = 0 And Len(sDestination) > 0 Then
        Exit Function
    End If
    Exit Function
errHandler:
```

```

    MsgBox "Error:DeleteMapping: " & Err.Description
End Function

Public Function RemoveAuth(sPostoffice As String, sUser As String)
    'returns the password for a user
    'this is located in the POPAccounts.tab file
    Dim oAUTHLogin As Object
    Dim IResult As Long
    Set oAUTHLogin = CreateObject("MEAOAU.Login")
    With oAUTHLogin
        .Account = sPostoffice
        .Password = ""
        .Status = -1
        .UserName = sUser
        .Description = ""
        IResult = .RemoveLogin()
    End With
    Exit Function
errHandler:
    'file problem usually..
    MsgBox "Could not delete auth entry."
End Function

Public Function RemoveDomain(sPostoffice As String, sDomain As String)
    On Error GoTo errHandler
    Dim oSMTPDomain As Object
    Dim IResult As Long
    Set oSMTPDomain = CreateObject("MEAOSM.Domain")
    With oSMTPDomain
        .AccountName = sPostoffice
        .DomainName = sDomain
        .DomainRedirectionHosts = ""
        .DomainRedirectionStatus = -1
        .Status = -1
        IResult = .RemoveDomain()
    End With
    Exit Function
errHandler:
    MsgBox "RemoveDomain:Error:" & Err.Description
End Function

Public Function GetMailRoot()
    GetMailRoot = GetRegistryString("SOFTWARE\mail enable\mail enable\connectors\srf", "Mail Root Directory")
End Function

Public Function GetMailConfigDirectory() As String
    On Error Resume Next
    GetMailConfigDirectory = GetRegistryString("SOFTWARE\Mail Enable\Mail Enable", "Configuration Directory")
End Function

```

10 More Information

More Information on MailEnable Developer Resources can be found at:

<http://www.mailenable.com/developers>

You can also access the MailEnable developers forum at:

<http://forum.mailenable.com/viewforum.php?f=4>