Codestone Ltd

Internet Mail Client Control Library

Welcome to the Codestone Internet Mail Client Control Library (CSMail), we hope you will find the library to be a rapid and painless way to add email functionality to your application.

With CSMail you can send and receive email using SMTP and POP3 – the standard email protocols on the Internet. Your emails can be as simple as a plain text message or as complex as a richly formatted HTML message with embedded stationary – and you can add as many attachments to the mail as you please.

By utilising ActiveX COM technology the CSMail library is available to programmers working in Visual Basic ®, Visual C++ ®, Microsoft® Office, ASP on IIS, the Windows Scripting Host and many more environments. Our customers have used CSMail to great effect in applications including ISS Web-Mail sites, email gateways and bespoke email clients.

Message Object Developer's Reference

SaveToFile Method

Declaration: SaveToFile(*Filename* as String)

Parameters

Filename	*The full pathname of the destination file.*

Description

Saves the entire message structure, including any attachments, to a disk file. This method saves the message in an optimised format that can only be read by the LoadFromFile method.

Return Value

There is no return value from this method. If any error occurs while saving the message an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

LoadFromFile Method

Declaration: LoadFromFile(*Filename* as String)

Parameters

Filename	*The full pathname of the source file.*

Description

Loads the entire message structure, including any attachments, from a disk file. This method can only load messages in the optimised format written by the SaveToFile method.

Return Value

There is no return value from this method. If any error occurs while loading the file an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Subject Property

Declaration: Subject as String

Description

The subject of the message. This property corresponds to the MIME 'Subject' header field.

```
MyMsg.Subject="Medical Officer's Report "
```

And

```
myMsg.Header("Subject")="Medical Officer's Report"
```

Are functionally equivalent.

Example

```
MyMsg.Subject="Medical Officer's Report"
```

From Collection Property

Declaration: From as AddressList

Description

The originators of the message.

This read/write property is a one-based collection of address. Developers will typically set the elements of the list before sending a message with the SMTPClient Object - or read the elements of the array after receiving a message with the POP3Client Object.

This property corresponds to the MIME 'From' header field:

```
MyMsg.From(1)="bones@enterprise.fed"
MyMsg.From(2)="spock@enterprise.fed"
```

And

```
myMsg.Header("From")="bones@enterprise.fed,spock@enterprise.fed"
```

Are functionally equivalent.

Addresses should be added to the list in consecutive order, starting at an index of one.

Care must be taken to ensure that a legal combination of 'From' and 'Sender' fields exists before the message is sent otherwise the SMTPClient will report an error - this table shows the allowed options.

Number of 'From' addresses	Number of 'Sender' Addresses	Notes
0	1	Sender is compulsory if there are no 'From' addresses.
1	0 or 1	Sender is optional if there is exactly one 'From' address.
>1	1	Sender is compulsory if there are more than one 'From' addresses.

The following formats are allowed for the specification of the From property:

Type	Example
Mailbox	Kirk@enterprise.fed or James T Kirk <kirk@enterprise.fed>

This property also supports the standard collection methods and properties (Add, Remove , Count and Item).

Example

```
MyMsg.From(1)="bones@enterprise.fed"
```

Sender Property

Declaration: Sender as AddressList

Description

The sender of the message.

This read/write property is a one-based collection of address. Developers will typically set the elements of the list before sending a message with the SMTPClient Object - or read the elements of the array after receiving a message with the POP3Client Object.

This property corresponds to the MIME 'Sender' header field:

```
MyMsg.Sender(1)="bones@enterprise.fed"
```

And

```
myMsg.Header("Sender")="bones@enterprise.fed"
```

Are functionally equivalent.

See the From Property for a description of the legal formats for email addresses for this property and for legal combinations of From and Sender addresses.

Example

```
MyMsg.Sender(1)="uhura@enterprise.fed"
```

ReplyTo Property

Declaration: ReplyTo as AddressList

Description

The address to which replies should be sent.

This read/write property is a one-based collection of address. Developers will typically set the elements of the list before sending a message with the SMTPClient Object - or read the elements of the array after receiving a message with the POP3Client Object.

This property corresponds to the MIME 'Reply-To' header field:

```
MyMsg.ReplyTo(1)="bones@enterprise.fed"
```

And

```
myMsg.Header("Reply-To")="bones@enterprise.fed"
```

Are functionally equivalent.

Addresses should be added to the list in consecutive order, starting at an index of one.

The following formats are allowed for the specification of the ReplyTo property:

Type	Example

Mailbox	`Kirk@enterprise.fed`
	or
	`James T Kirk <kirk@enterprise.fed>`

Group	`captains : kirk@enterprise.fed, janeway@voyager.fed;`

This property also supports the standard collection methods and properties (Add, Remove , Count and Item) together with the Visual Basic for .. each construct.

Example

```
MyMsg.ReplyTo(1)="bones@enterprise.fed"
```

To Collection Property

Declaration: To as AddressList

Description

The primary recipients of the message.

This read/write property is a one-based collection of address. Developers will typically set the elements of the list before sending a message with the SMTPClient Object - or read the elements of the array after receiving a message with the POP3Client Object.

This property corresponds to the MIME 'To' header field:

```
MyMsg.To(1)="bones@enterprise.fed"
```

And

```
myMsg.Header("To")="bones@enterprise.fed"
```

Are functionally equivalent.

Addresses should be added to the list in consecutive order, starting at an index of one.

The following formats are allowed for the specification of the To property:

Type	Example

Mailbox	`Kirk@enterprise.fed`
	or
	`James T Kirk <kirk@enterprise.fed>`

Group	`captains : kirk@enterprise.fed, janeway@voyager.fed;`

This property also supports the standard collection methods and properties (Add, Remove , Count and Item).

Example

```
MyMsg.To(1)="bones@enterprise.fed"
```

CC Collection Property

Declaration: CC as AddressList

Description

The secondary (CC) recipients of the message. This property corresponds to the MIME 'CC' header field.

Addresses should be added to the list in consecutive order, starting at an index of one.

See the To Property for a description of the legal formats for email addresses for this property.

This property also supports the standard collection methods and properties (Add, Remove , Count and Item).

Example

```
MyMsg.CC(1)="bones@enterprise.fed"
```

BCC Collection Property

Declaration: BCC as AddressList

Description

The tertiary (BCC) recipients of the message. The message is delivered to these recipients but is not included in the list of recipients.

This property corresponds to the MIME 'BCC' header field.

Addresses should be added to the list in consecutive order, starting at an index of one.

See the To Property for a description of the legal formats for email addresses for this property.

This property also supports the standard collection methods and properties (Add, Remove , Count and Item).

Example

```
MyMsg.BCC (1)="bones@enterprise.fed"
```

Sections Collection Property

Declaration: Sections as SectionCollection

Description

The sections (parts) of the message.

Each part of a message is held in a separate section. Simple messages may contain only a single section with a textual body, more complex messages may contain several sections including, for example, text and images.

Sections should be added to the list in consecutive order, starting at an index of one.

See the Sections Object reference for a full description of the Section Object.

This property also supports the standard collection methods and properties (Add, Remove , Count and Item).

Example

```
MyMsg.Sections(1).Body="Its life Jim, but not as we know it."
MyMsg.Sections(2).AttachBodyFromFile("tribbles.jpg")
```

Header Dictionary Property

Declaration: Header as Header

Description

The MIME Header for the message. This property provides access to the raw MIME header fields for the message.

The fields in the MIME header are automatically maintained as the properties of the object are changed and most applications will not need to access the headers directly.

The Header property is implemented as dictionary -type collection of strings and is keyed by the field name.

Example

```
MyMsg.Sections(1).Header("Content-Type")="multipart/alternative"
```

Tag Property

Declaration: Tag as String

Description

A user defined string value

UIDL Property

Declaration: UIDL as String

Description

The message UIDL, as reported bye the POP3 server.

This property is valid only for messages received with the POP3Client Object.

ByteSize Property

Declaration: ByteSize as long

Description

The message byte size, as reported by a POP3 server.

This property is valid only for messages received with the POP3Client Object.

Ordinal Property

Declaration: Ordinal as long

Description

The ordinal number of the message on the POP3 server.

This property is valid only for messages received with the POP3Client Object.

Section Object Developer's Reference

AttachBodyFromFile Method

Declaration: AttachBodyFromFile(*Filename* as String)

Parameters

Filename	*The full pathname of the source file.*

Description

Sets the section body to the contents of a file and marks the section as an attachment.

Return Value

There is no return value from this method. If any error occurs while attaching the file an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

ReadBodyFromFile Method

Declaration: ReadBodyFromFile(*Filename* as String)

Parameters

Filename	*The full pathname of the source file.*

Description

Sets the section body to the contents of a disk file and marks the section as 'inline'.

Return Value

There is no return value from this method. If any error occurs while reading the file an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

WriteBodyToFile Method

Declaration: WriteBodyToFile(*Filename* as String)

Parameters

Filename	*The full pathname of the destination file.*

Description

Writes the section body to a disk file.

Return Value

There is no return value from this method. If any error occurs while writing the file an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Body Property

Declaration: Body as String

Description

The section body.

This read/write property allows access to the text or data held in the body of the section - this property always reflects the native (un-encoded) form of the data - the data is automatically encoded when it is sent by SMTP client and decoded when it is received by the POP3 Client.

The Body property can only be used safely for accessing text content - binary data may be subject to undesired transformations - this is due to the need to convert from the character set used in message transmission to the Unicode character set used by OLE automation. The ReadBodyFromFile, AttachBodyFromFile and WriteBodyToFile methods provide a convenient means to transfer arbitrary binary data to and from disk files. The BodyBinary property provides a convenient method to access the binary contents of the section.

Example

```
MyMsg.Sections(1).Body="Its life Jim, but not as we know it."
```

BodyBinary Property

Declaration: BodyBinary as Variant

Comaptability: V 1.5.5 and greater

Description

The binary contents of the section body.

This read/write property allows access to the binary data held in the body of the section - this property always reflects the native (un-encoded) form of the data - the data is automatically encoded when it is sent by SMTP client and decoded when it is received by the POP3 Client.

The BodyBinary property can be used to safely access binary data without the undesirable transformations which conversion to and from unicode can produce.

Example (ASP)

```
Response.ContentType = section.Header("Content-Type")
Response.addHeader "content-disposition", "filename=" & section.filename
response.binarywrite (section.bodybinary)
```

Header Property

Declaration: Header as Header

Description

The MIME Header for the section.

This property provides access to the raw MIME header fields for the message.

Each section has it's own header which contains the fields which describe the content and type of the section.

The fields in the MIME header are automatically maintained as the properties of the object are changed and most applications will not need to access the headers directly.

The Header property is implemented as dictionary-type collection of strings and is keyed by the field name.

Example

```
myMsg.Header("Subject")="Message from Starfleet Headquarters"
```

Sections Collection Property

Declaration: Sections as SectionCollection

Description

The sub-sections (parts) of the section

RFC 1341 allows for multipart messages to contain nested multipart sections. This is supported by the Message Object Model and implemented in the SMTP and POP3 clients.

Sections should be added to the list in consecutive order, starting at an index of one.

Example

See Example 9 Nested multipart messages for an example of creating nested multipart messages.

CharacterSet Property

Declaration: CharacterSet as String

Description

The character set assigned to the section.

This property corresponds to the charset parameter of the RFC 1341 Content-Type field.

This property is set automatically by the control when either creating a new section or retrieving a message with the POP3Client Object. Applications may wish to examine this property after a message has been received in order to take appropriate action. Only advanced applications should attempt to set this property before sending a message.

Disposition Property

Declaration: Disposition as String

Description

The disposition type assigned to the section

This property relates to the disposition parameter of the RFC 2183 Content-Disposition field.

This property is set automatically by the control when either creating a new section or retrieving a message with the POP3Client Object. Applications may wish to examine this property after a message has been received in order to take appropriate action. Only advanced applications should attempt to set this property before sending a message.

In general the Disposition property will be either 'inline' or 'attachment' indicating a section which is to be included in the body or attached respectively.

Encoding Property

Declaration: Encoding as String

Description

The transfer encoding scheme of the section

The encoding schema of the body. May be `7bit`, `8bit`, `base64` or `quoted-printable`.

This property is set automatically by the control when either creating a new section or retrieving a message with the POP3Client Object. Applications may wish to examine this property after a message has been received in order to take appropriate action. Only advanced applications should attempt to set this property before sending a message.

Filename Property

Declaration: Filename as String

Description

The filename assigned to the section, may be empty

This property relates to the filename parameter of the RFC 2183 Content-Disposition field.

This property is set automatically by the control when either creating a new section or retrieving a message with the POP3Client Object. Applications may wish to examine this property after a message has been received in order to take appropriate action. Only advanced applications should attempt to set this property before sending a message.

Type Property

Declaration: Type as String

Description

The MIME type assigned to the section

This property relates to the type component of the RFC 1341 Content-Type field.

This property is set automatically by the control when either creating a new section or retrieving a message with the POP3Client Object. Applications may wish to examine this property after a message has been received in order to take appropriate action. Only advanced applications should attempt to set this property before sending a message.

SubType Property

Declaration: SubType as String

Description

The MIME sub-type assigned to the section

This property relates to the sub-type component of the RFC 1341 Content-Type field.

This property is set automatically by the control when either creating a new section or retrieving a message with the POP3Client Object. Applications may wish to examine this property after a message has been received in order to take appropriate action. Only advanced applications should attempt to set this property before sending a message.

SMTPClient Object Developer's Reference

SetProxyInfo Method

Declaration: SetProxyInfo(*ProxyInfo* as Object)

Parameters:

ProxyInfo	*An object providing information required to connect to a server through a proxy server. Currently allowed objects are:* ○ SOCK4ProxyInfo object ○ SOCK5ProxyInfo object ○ POP3ProxyInfo object ○ SMTPProxyInfo object

Description

Informs the SMTPClient or POP3Client object about the proxy/firewall connection required by the caller.

Return Value

There is no return value from this method. If any error occurs while a message is being sent an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Example

See the examples for the SOCK4ProxyInfo, SOCK5ProxyInfo, POP3ProxyInfo and SMTPProxyInfo objects

Connect Method

Declaration: Connect(*Host* as String,[*Port* as long])

Parameters

Host	*The address of the SMTP smart host. The host may be specified as a dotted IP address (e.g. 127.0.0.1) or as a host name (e.g. mail.myisp.com)*
Port	*The port on which to make the connection, if Port is not specified the connection will be made on port 25, the default for the SMTP protocol.*

Description

Attempts to connect to the named SMTP server using the specified port.

Return Value

There is no return value from this method. If any error occurs during the connection process an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

During the connection process the object will:

- Resolve the host name and attempt to create a TCP connection to the host.
- If the connection is successful the object will attempt to issue the preliminary HELO SMTP command and process the response.

ConnectESMTP Method

Declaration: ConnectESMTP(*Host* as String,[*Username* as String],[*Password* as String],[*Auth*(=LOGIN) as String],[*Port* as long])

Parameters

Host	*The address of the SMTP smart host. The host may be specified as a dotted IP address (e.g. 127.0.0.1) or as a host name (e.g. mail.myisp.com)*
Username	*The authentication username*
Password	*The authentication password*
Auth	*The authentication method. Currently only LOGIN is supported*
Port	*The port on which to make the connection, if Port is not specified the connection will be made on port 25, the default for the SMTP protocol.*

Description

Attempts to connect to the named ESMTP server using the specified authentication details and port.

Return Value

There is no return value from this method. If any error occurs during the connection process an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

During the connection process the object will:

- Resolve the host name and attempt to create a TCP connection to the host.
- If the connection is successful the object will attempt to issue the preliminary ESMTP EHLO and AUTH commands.

Close Method

Declaration: Close

Parameters

Description

Close the connection to the remote server

Return Value

There is no return value from this method. If any error occurs during the disconnection process an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

During the disconnection process the object will:

- Issue the SMTP QUIT command
- Close the TCP connection and release the socket.

SendMessage Method

Declaration: SendMessage(*Message* as Message)

Parameters

Message	*A message object. See the Message Object reference for details of the Message Object.*

Description

Send a message though the SMTP server. Connect () or ConnectESMTP () must be called before this method is called. SendMessage () may be called repeatedly before Close () is called if there are multiple messages to be sent.

Return Value

There is no return value from this method. If any error occurs while a message is being sent an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

While sending a message the object will:

- Issue the SMTP RCPT TO commands for each address in the Message Object's To and CC property array. RCPT TO commands are not issued for addresses in the BCC property array.
- Issue an SMTP MAIL FROM command for the appropriate originator field. If there is a single address in the Message Object's From property array then that address will be used. If there is more than one address in the Message Object's From property array then there must be a single Sender address.
- Issue an SMTP DATA command and transmit the message body to the SMTP server, encoding attachments appropriately if required.

SendMessageEx Method

Declaration: SendMessageEx(*Message* as IMessage, *EnvelopeTo* as String [,*EnvelopeFrom* as String])

Parameters

Message	*A message object. See the Message Object reference for details of the Message Object.*
EnvelopeTo	*The addresses to which the messages should be sent.*
EnvelopeFrom	*The address from which the message should be sent.*

Description

Send a message though the SMTP server. Connect () or ConnectESMTP () must be called before this method is called. SendMessageEx () may be called repeatedly before Close () if there are multiple messages to be sent.

Return Value

There is no return value from this method. If any error occurs while a message is being sent an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

While sending a message the object will:

- Issue the SMTP RCPT TO commands for each address in the list provided in the EnvelopeTo parameter.

- Issue an SMTP MAIL FROM command for the address in the EnvelopeFrom parameter. There should be exactly one address in this list. If there is not exactly one address in the list an error will be generated.

Issue an SMTP DATA command and transmit the message body to the SMTP server, encoding attachments appropriately if required.

LastServerResponseCode Property

Declaration: LastServerResponseCode as String

Description

The last SMTP response code from the remote server

Advanced

The property represents the numeric portion of the SMTP server's response (the 'response code') to the last command issued. Sophisticated applications may wish to examine this property in order to take corrective actions after an SMTP Error (Error number 80040206 or 80040204) has occurred. Developers should refer to Appendix E of RFC 821 for a detailed discussion of the theory of SMTP reply codes.

LastServerResponseString Property

Declaration: LastServerResponseString as String

Description

The last SMTP response from the remote server

Advanced

The property contains the SMTP server's response to the last command issued. Applications may wish to log or report this string to the user after an SMTP Error (Error number 80040206 or 80040204) has occurred.

LastSocketError Property

Declaration: LastSocketError as String

Description

The last error reported by the socket layer

Advanced

The property contains a description of the last error reported by the Winsock DLL. Applications may wish to inspect this property after a Socket Error (Error number 80040204 or 80040207) has occurred.

POP3 Client Developer's Reference

SetProxyInfo Method

Declaration: SetProxyInfo(*ProxyInfo* as Object)

Parameters:

ProxyInfo	*An object providing information required to connect to a server through a proxy server. Currently allowed objects are:* ○ SOCK4ProxyInfo object ○ SOCK5ProxyInfo object ○ POP3ProxyInfo object ○ SMTPProxyInfo object

Description

Informs the SMTPClient or POP3Client object about the proxy/firewall connection required by the caller.

Return Value

There is no return value from this method. If any error occurs while a message is being sent an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Example

See the examples for the SOCK4ProxyInfo, SOCK5ProxyInfo, POP3ProxyInfo and SMTPProxyInfo objects.

Connect Method

Declaration: Connect(*Host* as String, *Username* as String, *Password* as String,[*Port* as long])

Parameters

Host	*The address of the POP3 server. The host may be specified as a dotted IP address (e.g. 127.0.0.1) or as a host name (e.g. mail.myisp.com)*
Username	*The authentication username*
Password	*The authentication password*
Port	*The port on which to make the connection, if Port is not specified the connection will be made on port 110, the default for the POP3 protocol.*

Description

Attempts to connect to the named POP3 server using the specified port. After a successful connection is established the Messages collection property will contain an empty Message Object for each message on the maildrop, applications should subsequently call RetrieveMessages (), POP3RETR () or POP3TOP() to retrieve full or partial copies of the message.

Return Value

There is no return value from this method. If any error occurs during the connection process an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

During the connection process the object will:

- Resolve the host name and attempt to create a TCP connection to the host.

- If the connection is successful the object will attempt to issue the preliminary POP3 USER and PASS command
- If the authentication parameters are accepted the object issues a STAT command to establish the number of the messages in the maildrop and the total size of the messages. The object then populates the Messages collection property with an empty Message Object for each message.
- Issue a POP3 LIST command and set the ByteSize property of each Message Object in the collection appropriately.

Close Method

Declaration: Close([*CommitChanges*(=True) as long])

Parameters

CommitChanges *See description below.*

Description

Close the connection to the remote server. If CommitChanges is True (the default) then messages marked for deletion (see RetrieveMessages () and POP3DELE ()) are removed from the server, otherwise messages marked for deletion are not removed.

Return Value

There is no return value from this method. If any error occurs during the disconnection process an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

During the disconnection process the object will:

- Issue the POP3 RSET command if changes are not to be commited
- Issue the POP3 QUIT command
- Close the TCP connection and release the socket.

RetrieveMessages Method

Declaration: RetrieveMessages([*Index*(=0) as long],[*Lines*(=0) as long])

Parameters

Index	*Message Index. The default is all messages on the maildrop/*
Lines	*Number of lines to retrieve. The default is all message lines.*

Description

This is a high level method that encompasses the actions of many of the low level POP3 commands.

This method is suitable for applications with simple message processing requirements. More advanced applications should use the lower level POP3 commands for message processing.

Called with no parameters (or with both parameters as zero) RetrieveMessages () will populate all the elements of the Messages collection property with the messages from the maildrop and mark the messages for deletion from the server. This usage is designed to fulfill the requirements of a majority of simple applications.

Called with a non-zero Index parameter the object will retrieve the requested number of lines of the specified message from the maildrop and populate the corresponding element of the Messages collection property accordingly. If the number of lines requested is zero the entire message will be retrieved and the message will be marked for deletion from the server.

If partial messages are retrieved no attempt is made to decode the message or to split it into sections and the message is not marked for deletion from the server.

Return Value

There is no return value from this method. If any error occurs while messages are being retrieved an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

While retrieving messages the object will:

- Issue a POP3 RETR command (if the entire message is to be retrieved) or TOP command (if a given number of lines are required) for each appropriate message.
- Issue a POP3 DELE command for each message for which the RETR command was issued. DELE commands are not issued for commands retrieved with the TOP command.

POP3STAT Method (Low Level)

Declaration: POP3STAT

Return Value

There is no return value from this method. If any error occurs while the POP3 command is being executed an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

Issues a POP3 STAT command.

After issuing the STAT command the object parses the response from the server and sets the ByteCount property (of the POP3Client object) appropriately.

POP3LIST Method (Low Level)

Declaration: POP3LIST([*Index*(=0) as long])

Parameters

Index	*Message Index. The default is zero.*

Return Value

There is no return value from this method. If any error occurs while the POP3 command is being executed an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

Issues a POP3 LIST command.

If the Index parameter is zero then the LIST command is issued with no parameters and the POP server returns information for all the messages in the maildrop.

If the Index parameter is non-zero the Index is used as the parameter to the command and the POP server returns information about the specified message.

After issuing the LIST command the object parses the response from the server and sets the ByteSize property of the appropriate Message Object or Objects accordingly.

POP3UIDL Method (Low Level)

Declaration: POP3UIDL([*Index*(=0) as long])

Parameters

Index	*Message Index. The default is zero.*

Return Value

There is no return value from this method. If any error occurs while the POP3 command is being executed an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

Issues a POP3 UIDL command.

If the Index parameter is zero then the UIDL command is issued with no parameters and the POP server returns information for all the messages in the maildrop.

If the Index parameter is non-zero the Index is used as the parameter to the command and the POP server returns information about the specified message.

After issuing the UIDL command the object parses the response from the server and sets the UIDL property of the appropriate Message Object or Objects accordingly.

POP3TOP Method (Low Level)

Declaration: POP3TOP(*Index* as long, *Lines* as long)

Parameters

Index	*Message Index.*
Lines	*The number of lines to retrieve*

Return Value

There is no return value from this method. If any error occurs while the POP3 command is being executed an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

Issues a POP3 TOP command using the provided parameters.

After issuing the TOP command the object parses the response from the server and populates the Header property of the appropriate Message Object. The body of the messages is set to the required number of lines from the message. No attempt is made to decode or explode the message sections.

If the zero is specified for the Lines parameter only the message headers are retrieved.

POP3RETR Method (Low Level)

Declaration: POP3RETR(*Message* as long)

Parameters

Index	*Message Index.*

Return Value

There is no return value from this method. If any error occurs while the POP3 command is being executed an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

Issues a POP3 RETR command.

After issuing the RETR command the object parses the response from the server and populates the Header property of the appropriate Message Object. The body is decoded and the sections are exploded into the Sections collection property of the appropriate Message Object.

POP3DELE Method (Low Level)

Declaration: POP3DELE(*Message* as long)

Parameters

Index	*Message Index.*

Return Value

There is no return value from this method. If any error occurs while the POP3 command is being executed an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

Issues a POP3 DELE command. The POP3 server marks the message as deleted. Any future reference to the message-number associated with the message in a POP3 command generates an error.

Messages marked for deletion are not removed from the server until the connection is closed with a call to either Close (True) or POP3QUIT ().

POP3NOOP Method (Low Level)

Declaration: POP3NOOP

Return Value

There is no return value from this method. If any error occurs while the POP3 command is being executed an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

Issues a POP3 NOOP command. The POP3 server does nothing, it merely replies with a positive response.

POP3RSET Method (Low Level)

Declaration: POP3RSET

Return Value

There is no return value from this method. If any error occurs while the POP3 command is being executed an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

Issues a POP3 RSET command. The POP3 server unmarks any messages marked for deletion.

POP3Client.POP3QUIT Method (Low Level)

Declaration: POP3QUIT

Return Value

There is no return value from this method. If any error occurs while the POP3 command is being executed an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Advanced

Issues a POP3 QUIT command. The POP3 server removes all messages marked as deleted from the maildrop.

ByteCount Property

Declaration: ByteCount as long

Description

The byte count of all the messages on the POP server, as reported by the server. This property is set automatically during a call to Connect () or POP3STAT ().

POP3Client.LastServerResponseString Property

Declaration: LastServerResponseString as String

Description

The last POP3 response from the remote server

Advanced

The property contains the POP3 server's response to the last command issued. Applications may wish to log or report this string to the user after a POP3 Error (Error number 8004020B) has occurred.

LastSocketError Property

Declaration: LastSocketError as String

Description

The last error reported by the socket layer

Advanced

The property contains a description of the last error reported by the Winsock DLL. Applications may wish to inspect this property after a Socket Error (Error number 80040207) has occurred.

MessageCount Property

Declaration: MessageCount as long

Description

The count of all the messages in the maildrop on the remote server, as reported by the server. This property is set automatically during a call to Connect () or POP3STAT ().

Messages Collection Property

Declaration: Messages as MessageCollection

Description

A collection of Messages Objects representing the messages on the remote server. The Message Objects will be empty until an appropriate call to RetrieveMessages (), POP3RETR () or

POP3TOP as been made. *See the Message Object reference* for details of the Message Object.

FileClient Object Developer's Reference

The FileClient object allows developers to read and write messages stored in MIME format.

Saving and, particularly, reading, messages in MIME format on disk is not an efficient process. Developers should only use the methods of the FileClient Object where MIME format files are specifically required by interoperability requirements.

In situations where interoperability requirements to do not require MIME format files developers are encouraged to use the LoadFromFile() and SaveToFile() methods of the MessageObject – these methods use an optimised format to store the message data and will result in better performance.

Read Method

Declaration: Read(*Filepath* as String) as Message

Parameters

Filepath	*Pathname of file.*

Return Value

Returns the message read from the file.

Description

Reads a message from a file containing a single MIME format message.

Write Method

Declaration: Write(*Filepath* as String, *Message* as Message)

Parameters

Filepath	*Pathname of file.*
Message	*Message to write*

Return Value

There is no return value from this method. If any error occurs while the command is being executed an error will be raised and should be handled through the VB/VBA/VBScript On Error mechanism.

Description

Writes single message to a file, in MIME format.

ProxyInfo Objects Developer's Reference

CSMail provides a consistent model for proxy and firewall support. This approach is achieved through the SetProxyInfo methods on the SMTPClient and POP3Client objects together with an object for each firewall/proxy methodology. The methodology specific objects will allow us to extend the firewall/proxy support without changing the interfaces on the main client objects.

SOCK4ProxyInfo Object

Provides configuration information for a SOCKS V4 proxy server.

Developers wishing to provide SOCKS 4 support in their application should provide a means (typically a dialog box or configuration file) for the user to provide the information necessary to connect to a SOCKS 4 server. Developers may wish to advise end users to consult their network administrators for this information.

ProxyAddress Property

Declaration: ProxyAddress as String

Description

The address of the of the SOCKS 4 proxy server. *The host may be specified either as a dotted IP address (e.g. 127.0.0.1) or as a host name (e.g. socks4.mydomain.com)*

ProxyPort Property

Declaration: ProxyPort as long

Description

The port on which to connect to the proxy server. The default is 1080 (the standard SOCKS proxy server port).

ProxyUserID Property

Declaration: ProxyUserID as String

Description

The Socks 4 UserID.

Example

```
' ------------------------------------------
' Create SOCK4ProxyInfo object and set values
' In production code these values will come
' from the user.
' ------------------------------------------

Set S4=CreateObject("CSMail.SOCK4ProxyInfo")
S4.ProxyAddress="sock4.mydomain.com"
' (The library will use sensible defaults for ProxyPort and ProxyUserId)


' ------------------------------------------
' Create POP3Client object
' ------------------------------------------
Set pop=CreateObject("CSMail.Pop3Client")

' ------------------------------------------
' Pass Proxy information to client
' ------------------------------------------
call pop.SetProxyInfo(S4)

' ------------------------------------------
' Connect to the pop server.
' ------------------------------------------
call pop.Connect("pop3.mydomain.com","username","secret")
```

SOCK5ProxyInfo Object

Provides configuration information for a SOCKS V5 proxy server.

Developers wishing to provide SOCKS 5 support in their application should provide a means (typically a dialog box or configuration file) for the user to provide the information necessary to connect to a SOCKS 5 server. Developers may wish to advise end users to consult their network administrators for this information.

ProxyAddress Property

Declaration: ProxyAddress as String

Description

The address of the of the SOCKS 5 proxy server. *The host may be specified either as a dotted IP address (e.g. 127.0.0.1) or as a host name (e.g. socks5.mydomain.com)*

ProxyPort Property

Declaration: ProxyPort as long

Description

The port on which to connect to the proxy server. The default is 1080 (the standard SOCKS proxy server port).

ProxyUsername Property

Declaration: ProxyUsername as String

Description

The Socks 5 Username.

ProxyPassword Property

Declaration: ProxyPassword as String

Description

The Socks 5 Password.

Example

```
' ------------------------------------------------
' Create SOCK5ProxyInfo object and set values
' In production code these values will come
' from the user.
' ------------------------------------------------

Set S5=CreateObject("CSMail.SOCK5ProxyInfo")
S5.ProxyAddress="sock5.mydomain.com"
S5.ProxyPort=1080
S5.ProxyUsername="proxyuser"
S5.ProxyPassword="proxysecret"


' ------------------------------------------------
' Create POP3Client object
' ------------------------------------------------
Set pop=CreateObject("CSMail.Pop3Client")

' ------------------------------------------------
' Pass Proxy information to client
```

```
' -----------------------------------------
call pop.SetProxyInfo(S5)

' -----------------------------------------
' Connect to the pop server.
' -----------------------------------------
call pop.Connect("pop3.mydomain.com","username","secret")
```

POP3ProxyInfo Object

Provides configuration information for a standard pop3 proxy server.

Developers wishing to provide pop3 proxy support in their application should provide a means (typically a dialog box or configuration file) for the user to provide the information necessary to connect to a pop3 proxy server. Developers may wish to advise end users to consult their network administrators for this information.

ProxyAddress Property

Declaration: ProxyAddress as String

Description

The address of the of the pop3.proxy server. *The host may be specified either as a dotted IP address (e.g. 127.0.0.1) or as a host name (e.g. popproxy.mydomain.com)*

ProxyPort Property

Declaration: ProxyPort as long

Description

The port on which to connect to the proxy server. *The normal port for connection to a pop3.proxy server is 110, the same as true POP3 server.*

ProxySeperatorChar Property

Declaration: ProxySeperatorChar as String

Description

Seperator character

Example

```
' ------------------------------------------------
' Create POP3ProxyInfo object and set values
' In production code these values will come
' from the user.
' ------------------------------------------------

Set P3Proxy=CreateObject("CSMail.POP3ProxyInfo")
P3Proxy.ProxyAddress="pop3proxy.mydomain.com"
P3Proxy.ProxyPort=110

' ------------------------------------------------
' Create POP3Client object
' ------------------------------------------------
Set pop=CreateObject("CSMail.Pop3Client")

' ------------------------------------------------
' Pass Proxy information to client
' ------------------------------------------------
call pop.SetProxyInfo(P3Proxy)

' ------------------------------------------------
' Connect to the pop server.
' ------------------------------------------------
call pop.Connect("pop3.mydomain.com","username","secret")
```

SMTPProxyInfo Object

Provides configuration information for a standard smtp proxy server.

Developers wishing to provide smtp proxy support in their application should provide a means (typically a dialog box or configuration file) for the user to provide the information necessary to connect to a smtp.proxy server. Developers may wish to advise end users to consult their network administrators for this information.

ProxyAddress Property

Declaration: ProxyAddress as String

Description

The address of the of the smtp proxy server. *The host may be specified either as a dotted IP address (e.g. 127.0.0.1) or as a host name (e.g. popproxy.mydomain.com)*

ProxyPort Property

Declaration: ProxyPort as long

Description

The port on which to connect to the proxy server. *The normal port for connection to an smtp proxy server is 25, the same as true POP3 server.*

Example

```
' ------------------------------------------
' Create SMTPProxyInfo object and set values
' In production code these values will come
' from the user.
' ------------------------------------------

Set SMTPProxy=CreateObject("CSMail.SMTPProxyInfo")
SMTPProxy.ProxyAddress="smtp3proxy.mydomain.com"
SMTPProxy.ProxyPort=25

 ' ------------------------------------------
' Create SMTPClient object
' ------------------------------------------
Set smtp=CreateObject("CSMail.SMTPClient")

' ------------------------------------------
' Pass Proxy information to client
' ------------------------------------------
call smtp.SetProxyInfo(SMTPProxy)

' ------------------------------------------
' Connect to the pop server.
' ------------------------------------------
call smtp.Connect("smtp.mydomain.com")
```

RFC2047 Object

Provides a convenient mechanism for encoding and decoding international text according to RFC 2047 (MIME Part Three: Message Header Extensions for Non-ASCII Text).

Encoding Property

Declaration: Encoding as String

Description

Developers should set this property before calling the Encode method if they wish to explicitly define the encoding scheme to be used.

Developers can examine the value of this property after calling the Decode method to determine the encoding scheme that was used to encode the text.

Possible values are "Q" (for a quoted-printable like encoding) and "B" (for Base64 encoding). The default value is "Q".

Charset Property

Declaration: Charset as String

Description

Developers should set this property before calling the Encode method if they wish to explicitly define the character set used to encode the text.

If this property is set successfully the Codepage property will be automatically updated to the appropriate code page for the character set. If there is no appropriate code page for the character set then errNoCodepage will be raised. The code page / character set mapping is resolved by querying the Windows registry for character sets and code pages installed on the system.

Developers can examine the value of this property after calling the Decode method to determine the character set that was used to encode the text.

The default value is the character set corresponding to the active code page on the system.

Codepage Property

Declaration: Codepage as long

Description

Developers should set this property before calling the Encode method if they wish to explicitly define the codepage used to encode the text.

If this property is set successfully the Charset property will be automatically updated to the appropriate character set for the codepage. If there is no appropriate character set for the code page set then

errNoCharset will be raised. The code page / character set mapping is resolved by querying the Windows registry for character sets and code pages installed on the system.

Developers can examine the value of this property after calling the Decode method to determine the code page appropriate for displaying the text.

The default value is the active code page on the system.

Encode Method

Declaration: Encode(*Text as* String) *as* String

Parameters:

Text	*The text to be encoded*

Description

Encodes the text to a RFC 2047 string.

Return Value

The encoded text.

Decode Method

Declaration: Decode(*Text as* String) *as* String

Parameters:

Text	*The text to be decoded*

Description

decodes the text from a RFC 2047 string to a Unicode string.

Return Value

The decoded text.

Settings Object

Provides a mechanism for setting application-wide options for the CSMail library.

Option Property

Declaration: Option(*Option as* long) as Variant

Parameters:

Index	*The option identifier*

Description

Developers can use this property to set various application-wide options for the library.

Identifier	***Value***	***Notes***	***Default***
N/A	0-5	Reserved	N/A
optionRFC2047Unicode	6	True - the control will fully decode the text into a Unicode string with the characters represented as the correct Unicode equivalents. This is the default setting. False – multibyte characters (for example ideographs) will be converted to a single byte character string representation.	True
N/A	7	Reserved	N/A
optionDisableAutoMime	8	True - the control will **not** attempt to convert non-MIME messages to a MIME format. UUEncoded attachments in a RFC822 message body will **not** be converted to MIME sections. False –the control will convert non-MIME message to a MIME format. UUEncoded attachments in a RFC822 message body will be converted to MIME sections.	False
optionSocketLogFile	9	The pathname of a file to which the control will log conversations with POP3 and SMTP servers. An empty string disables socket logging. Logging socket activity will impact on performance and may consume large amounts of disk storage. Developers should only use this option for diagnostic purposes.	An empty string
optionLogData	10	Reserved	N/A
optionLibraryBuild	11	The current build number of the library. (Read Only).	N/A
optionLibraryPath	12	The path of the library. (Read Only).	N/A
optionReserved6	13	Reserved	N/A
optionReserved7	14	Reserved	N/A
optionQuickPop3Connect	15	If set to true – the POP3Client object will not issue STAT and LIST commands during the call to Connect().	False
optionDisableRFC1035Domains	16	Disable strict (RFC 1035) parsing of domains.	False

optionLaxParameterParsing	17	Allow some laxer parsing of the MIME parameter BNF production. (**Not recommended**.)	False
optionTxTimeout	18	Socket send timeout (milliseconds). **Note:** This option must be set before the respective SMTPClient or POP3Client object is instantiated. Setting the option will not alter the timeout for currently instantiated objects.	300000 (5 minutes)
optionRxTimeout	19	Socket receive timeout (milliseconds) **Note:** This option must be set before the respective SMTPClient or POP3Client object is instantiated. Setting the option will not alter the timeout for currently instantiated objects.	300000 (5 minutes)
optionConnectTimeout	20	Socket connect timeout (milliseconds) **Note:** This option must be set before the respective SMTPClient or POP3Client object is instantiated. Setting the option will not alter the timeout for currently instantiated objects.	300000 (5 minutes)

LicenseInfo Object

The LicenseInfo object allows developers of late bound applications to specify the location of the CSMail runtime license file.

Late bound applications are those that use the IDispatch interface to call methods in the library. VBA, vbscript and ASP are examples of development environments that produce late bound code. Compiled languages such as Visual Basic and C++ produce early bound code and developers using these environments do not need to use the LicenseInfo Object.

RuntimeLicenseFile Property

Declaration: RuntimeLicense as String

Description

Specifies the location of the CSMail runtime license file. At run time the library will look in this location for the license file.

Appendix A VBScript Examples

Example 1 Create a simple message and send it with the SMTPClient Object

```
' Example:  CSMail VBScript Example 1
' Summary:  Create a simple message and send it with the SMTPClient Object
' Usage:    cscript exvbs01.vbs
'           or
'           wscript exvbs01.vbs

set MyMsg=CreateObject("CSMail.Message")      ' Create the message object

MyMsg.Subject="Medical Officer's Report"      ' Set the message subject
MyMsg.To(1)="kirk@enterprise.com"             ' Set the recipient...
MyMsg.From(1)="bones@enterprise.com"          ' ... and the originator
MyMsg.Sections(1).Body="Its life Jim, but not as we know it."
                                              ' Set the message body text

set SMTP = CreateObject("CSMail.SMTPClient")  ' Create an SMTP Client Object
SMTP.Connect("127.0.0.1")                     ' Connect to the server -
 in this case the local machine
SMTP.SendMessage(MyMsg)                       ' Send the message
SMTP.Close                                    ' Close the server
```

Example 2 Create a message with a single file attachment and send it with SMTPClient Object

```
' Example:  CSMail VBScript Example 2
' Summary:  Create a message with a single file attachment and send it with the SMTPCl
ient Object
' Usage:    cscript exvbs02.vbs
'           or
'           wscript exvbs02.vbs

set MyMsg=CreateObject("CSMail.Message")      ' Create the message object

MyMsg.Subject="Medical Officer's Report"      ' Set the message subject
MyMsg.To(1)="kirk@enterprise.com"             ' Set the recipient...
MyMsg.From(1)="bones@enterprise.com"          ' ... and the originator
MyMsg.Sections(1).Body="Its life Jim, but not as we know it."
                                              ' Set the message body text

' Attach a file
' Note - the Content-Type and Content-Transfer-
Encoding fields will be set automatically
' by the control
MyMsg.Sections(2).AttachBodyFromFile("V:\test\tribbles.giff")


set SMTP = CreateObject("CSMail.SMTPClient")  ' Create an SMTP Client Object
SMTP.Connect("127.0.0.1")                     ' Connect to the server -
 in this case the local machine
SMTP.SendMessage(MyMsg)                       ' Send the message
SMTP.Close                                    ' Close the server
```

Example 3 Create a message containing all the files in a directory and send it with the SMTPClientObject

```
' Example:  CSMail VBScript Example 3
' Summary:  Create a message containing all the files in a directory and
'           send it with the SMTPClientObject
' Usage:    cscript exvbs03.vbs
'           or
'           wscript exvbs03.vbs

set MyMsg=CreateObject("CSMail.Message")      ' Create the message object

MyMsg.Subject="Medical Officer's Report"      ' Set the message subject
MyMsg.To(1)="kirk@enterprise.com"             ' Set the recipient...
MyMsg.From(1)="bones@enterprise.com"          ' ... and the originator
MyMsg.Sections(1).Body="Its life Jim, but not as we know it."
                                              ' Set the message body text

' See the Microsoft Windows Scripting Host Documentation for details
' of the FileSystemObject model
Set fso=CreateObject("Scripting.FileSystemObject")
Set folder=fso.GetFolder("v:\test")

for each file in folder.files                 ' Iterate through the files
  wscript.echo file.name                      ' Diagnostics (see WSH docs re wscript.e
cho)
  set s=MyMsg.Sections.Add                    ' Create a new section in the current me
ssage
  s.AttachBodyFromFile(file.Path)             ' Attach the file
  next


set SMTP = CreateObject("CSMail.SMTPClient")  ' Create an SMTP Client Object
SMTP.Connect("127.0.0.1")                     ' Connect to the server -
 in this case the local machine
wscript.echo "Sending..."
SMTP.SendMessage(MyMsg)                       ' Send the message
wscript.echo "Sent..."
SMTP.Close                                    ' Close the server
```

Example 4 Read all messages on a Pop3 Server and then save all the attachments

```
' Example:  CSMail VBScript Example 4
' Summary:  Read all messages on a Pop3 Server and then save all the attachments
' Usage:    cscript exvbs04.vbs
'           or
'           wscript exvbs04.vbs


set pop=CreateObject("csmail.Pop3client")             ' Create a Pop3Client Object
call pop.Connect("localhost","myusername","secret")
                                                      ' and connect to the server

wscript.echo "Retrieving Messages..."
call pop.RetrieveMessages                             ' Get all the messages on the se
rver
wscript.echo "Done..."

pop.Close(false)                                      ' Close the connection to the se
rver
                                                      ' (but don't delete the messages

)

' Now we iterate through all the messages

for each msg in pop.Messages                          ' The messages property is a col
lection of
                                                      ' all the messages we retrieved
  wscript.echo msg.Subject
  for each section in msg.Sections                    ' The sections property is a col
lection of
                                                      ' all the sections (parts) of th
e message
    if section.Filename<>"" then
      wscript.echo section.Filename                   ' echo the filename
      section.WriteBodyToFile("v:\attachments\"&section.Filename)
                                                      ' Save the attachment in a suita
```

```
ble place
      end if
    next
  next
```

Example 5 Read all messages on a Pop3 Server and then save all the messages to disk files

```
' Example:  CSMail VBScript Example 5
' Summary:  Read all messages on a Pop3 Server and then save all the messages to disk
files
' Usage:    cscript exvbs05.vbs
'           or
'           wscript exvbs05.vbs


set pop=CreateObject("csmail.Pop3client")             ' Create a Pop3Client Object
call pop.Connect("localhost","myusername","secret")
                                                      ' and connect to the server

wscript.echo "Retrieving Messages..."
call pop.RetrieveMessages                             ' Get all the messages on the se
rver
wscript.echo "Done..."

pop.Close(true)                                       ' Close the connection to the se
rver
                                                      ' (and delete the messages on th
e server)


' Now we iterate through all the messages

for each msg in pop.Messages                          ' The messages property is a col
lection of
                                                      ' all the messages we retrieved
  wscript.echo msg.UIDL
  msg.SaveToFile "v:\inbox\"&msg.UIDL                 ' Save the message to a suitable
 location
  next
```

Example 6 Reload previously saved messages

```
' Example:  CSMail VBScript Example 6
' Summary:  Load all the messages we saved in Example 5
' Usage:    cscript exvbs06.vbs
'           or
'           wscript exvbs06.vbs


' See the Microsoft Windows Scripting Host Documentation for details
' of the FileSystemObject model
Set fso=CreateObject("Scripting.FileSystemObject")
Set folder=fso.GetFolder("v:\inbox")

for each file in folder.files                 ' Iterate through the files
  set MyMsg=CreateObject("CSMail.Message")    ' Create the message object
  MyMsg.LoadFromFile(file.Path)
  wscript.echo MyMsg.Subject

  for each section in MyMsg.Sections                  ' The sections property is a col
lection of
                                                      ' all the sections (parts) of th
e message
    if section.Filename<>"" then
      wscript.echo section.Filename                   ' echo the filename
      section.WriteBodyToFile("v:\attachments\"&section.Filename)
                                                      ' Save the attachment in a suita
ble place
      end if
      next

  wscript.echo "Done!"

  next
```

Example 7 Using the Message.Header property

```
' Example:  CSMail VBScript Example 7
' Summary:  Using the Message.Header property
' Usage:    cscript exvbs07.vbs
'           or
'           wscript exvbs07.vbs

set MyMsg=CreateObject("CSMail.Message")       ' Create the message object

MyMsg.Header("X-ID")="NCC 1701"                ' Set a user-defined field

WScript.echo("The X-ID field contains: "&MyMsg.Header("X-ID"))
                                               ' Echo the user-defined field

' Show all the MIME message fields
WScript.echo("Dumping all MIME Fields:")
for each field in MyMsg.Header
  WScript.echo(field&"="&MyMsg.Header(field))
  next
```

Example 8 To, CC, BCC, ReplyTo, From, Sender and Subject Properties

```
' Example:  CSMail VBScript Example 8
' Summary:  To, CC, BCC, ReplyTo, From, Sender and Subject Properties
' Usage:    cscript exvbs08.vbs
'           or
'           wscript exvbs08.vbs

' 1) Set the fields the easy way - through the properties

WScript.echo("First method")

set MyMsg=CreateObject("CSMail.Message")       ' Create the message object

MyMsg.To(1)="kirk@enterprise.fed"              ' Add first primary recipient
MyMsg.To(2)="spock@enterprise.fed"             ' Add second primary recipient

MyMsg.CC(1)="checkov@enterprise.fed"           ' Add first copy recipient
MyMsg.CC(2)="sulu@enterprise.fed"              ' Add second copy recipient

MyMsg.BCC(1)="bones@enterprise.fed"            ' Add first blind copy recipient
MyMsg.BCC(2)="scotty@enterprise.fed"           ' Add second blind copy recipient

MyMsg.From(1)="uhura@enterpise.fed"            ' Could have multiple Froms if we wanted
MyMsg.Sender(1)="uhura@enterpise.fed"          ' Can only have one Sender
MyMsg.ReplyTo(1)="uhura@enterpise.fed"

MyMsg.Subject="Message from Starfleet Headquarters"

' Note -
 setting these properties automatically updates appropriate Fields in the Messge Heade
r collection
' Let's dump the Header to see what's happened-

' Show all the MIME message fields
WScript.echo("Dumping all MIME Fields:")
for each field in MyMsg.Header
  WScript.echo("--"&field&"="&MyMsg.Header(field))
  next

' 2) Set the properties the harder (?) way - through the Header OLE collection ...
WScript.echo("Second method")

set MyMsg=CreateObject("CSMail.Message")       ' Create  a new message object
myMsg.Header("To")="kirk@enterprise.fed,spock@enterprise.fed"
myMsg.Header("CC")="checkov@enterprise.fed,sulu@enterprise.fed"
myMsg.Header("BCC")="bones@enterprise.fed,scotty@enterprise.fed"
myMsg.Header("From")="uhura@enterpise.fed"
myMsg.Header("Sender")="uhura@enterpise.fed"
myMsg.Header("Reply-To")="uhura@enterpise.fed"
myMsg.Header("Subject")="Message from Starfleet Headquarters"

' Note - setting these properties automatically updates the appropriate properties
' Let's dump the properties to check...


WScript.echo("'To' recipients:")
```

```
for each address in MyMsg.To
  WScript.echo "--"+address
  next

WScript.echo("'CC' recipients:")
for each address in MyMsg.CC
  WScript.echo "--"+address
  next

WScript.echo("'BCC' recipients:")
for each address in MyMsg.BCC
  WScript.echo "--"+address
  next

' etc.....
```

Example 9 Nested multipart messages

```
' Notes:
' In this example we will create a nested multipart messages and show one
' technique you may use for accessing such a message
'
' A common real
life example of a  nested multipart message involves a message which has a text sectio
n
' represented in two alternative formats and a single attached file ie:
'
' Message multipart/mixed
'   Section1 multipart/alternative
'     Section11 [Normal Text]
'     Section12 [HTML]
'   Section2  [Attached File]
'


set MyMsg=CreateObject("CSMail.Message")                    ' Create the message objec
t


' Sections(1) has two sub-sections and the default Content-
Type for Section(1) would be
' multipart/mixed but we need it to be mulipart/alternative so we set the Mime Header
Field
' appropriately - in this case we MUST use the Header collection rather than the Type/
' SubType properties - try doing it the otherway and you'll see why!
'
' Note that the control will handle the boundary parameter for us which saves some has
sle!


MyMsg.Sections(1).Header("Content-Type")="multipart/alternative"


' Sections(1).Sections(1) - Content type will default to text/plain
MyMsg.Sections(1).Sections(1).Body="This is the plain text message."


' Sections(1).Sections(2) - Need to set Content-Type to text/html
' - this time we can use the properties.
MyMsg.Sections(1).Sections(2).Body="This is the &ltB&gt;html&lt/B&gt version."
MyMsg.Sections(1).Sections(2).Type="text"
MyMsg.Sections(1).Sections(2).SubType="html"


MyMsg.Sections(2).AttachBodyFromFile("V:\test\tribbles.giff")


' ... Could send this with the SMTPClient now...


' ... but we'll pretend we've just received the message with the POP3Client object and
 we need to
' process _all_ the sections


' This is how NOT to do it -  we miss  the sub-sections of Section(1)
wscript.echo "The wrong way!"
for each section in MyMsg.Sections
  wscript.echo "Processing section Content-Type is "&section.Header("Content-Type")
  next


' The easiest way is probably to use recursion -
 the ProcessSections subroutine declared
' at the bottom of the file does this
```

```
wscript.echo "One right way!"
ProcessSections(MyMsg.Sections)


sub ProcessSections(Sections)


for each section in Sections
  if (section.type<>"multipart") then
    wscript.echo "Recursively Processing section Content-
Type is "&section.Header("Content-Type")
    end if
  ProcessSections(section.Sections) ' Process sub-sections
  next


end sub
```

Appendix B CSMail Error Numbers and Messages

ErrSMTPFromSyntax	(80000200H)	48
errSMTPFromTooMany	(80000201H)	48
errSMTPSenderTooMany	(80000202H)	48
Unused	(80000203H)	48
ErrSMTPOrSocket	(80000204H)	48
errConnect	(80000205H)	48
errSMTP	(80000206H)	49
errSocket	(80000207H)	49
errFileNotFound	(80000208H)	49
errIndex	(80000209H)	49
errInvalidAddress	(8000020AH)	49
errPOP	(8000020BH)	49
errInternalParseField	(8000020CH)	49
errInvalidContentType	(8000020DH)	49
errAllreadyConnected	(8000020EH)	50
errBusySending	(8000020FH)	50
errSocketSending	(80000210H)	50
Unused	(80000211H)	50
errInvalidDisposition	(80000212H)	50
errESMTP	(80000213H)	50
errESMTP_AUTH	(80000214H)	50
errSOCKS4	(80000215H)	50
errSOCKS5	(80000216H)	51
errInvalidProxyType	(80000217H)	51
Unused	(80000218H-80000222)	51
errNoCodepage	(80000223H)	51
errNoCharset	(80000224H)	51
Unused	(80000225H-80000226)	51
errPOP3QuickConnectStat	(80000227H)	51
errPOP3ServerClosedPort	(80000228H)	51
errTimeout	(80000229H)	51

ErrSMTPFromSyntax (80000200H)

Raised by the SMTPClient Object if the syntax of the 'From' field is invalid. This is most likely to occur if you pass an invalid address in the *EnvelopeFrom* parameter of SendMessageEx method.

Diagnosis/Resolution:

- You, or a user of your application, have specified an invalid email address, correct the situation and retry the operation.

errSMTPFromTooMany (80000201H)

Raised by the SMTPClient Object if the number of 'From' and 'Sender' fields is invalid. The table below lists the valid combinations of From and Sender addresses.

Number of 'From' Addresses	Number of 'Sender' Addresses	Comment
0	1	Sender is compulsory if there are no 'From' addresses.
1	0 or 1	Sender is optional if there is exactly one 'From' address.
>1	1	Sender is compulsory if there are more than one 'From' addresses.

Diagnosis/Resolution:

- You, or a user of your application, have specified an invalid combination of email addresses, correct the situation and retry the operation.

errSMTPSenderTooMany (80000202H)

Raised by the SMTPClient Object if there is more than one address in the Sender field. See ErrSMTPFromTooMany for valid combinations of From and Sender addresses.

Diagnosis/Resolution:

- You, or a user of your application, have specified an invalid combination of email addresses, correct the situation and retry the operation.

Unused (80000203H)

This error number is currently unused.

ErrSMTPOrSocket (80000204H)

Raised by the SMTPClient Object if an error occurs. This error will be raised in the specific case where an incomplete response is received from the SMTP server.

Diagnosis/Resolution:

- The network or server has probably failed. Retry the operation at a later time.

errConnect (80000205H)

Raised by the SMTPClient and POP3Client objects when they are unable to establish a connection to the server.

Diagnosis/Resolution:

- You, or a user of your application, may have mistyped the name of the server, correct the situation and retry the operation.
- The server may be down. Retry the operation at a later time.

- The network may have failed. Retry the operation at a later time.
- The server may be protected by a firewall, implement proxy support using the Proxy objects.

errSMTP (80000206H)

Raised by the SMTP client when an error occurs in the SMTP protocol.

Diagnosis/Resolution:

- Examine the contents of the LastServerResponseCode and LastServerResponseString properties and take appropriate action before retrying the operation.

errSocket (80000207H)

Raised by the SMTPClient and POP3Client objects when a socket error occurs.

Diagnosis/Resolution:

- The network or server has probably failed. The error description includes a verbose explanation of the nature of the socket error that you may or may not wish to present to the user. Retry the operation at a later time.

errFileNotFound (80000208H)

Raised by the ReadBodyFromFile and WriteBodyToFile methods of the section object if the specified file cannot be opened.

Diagnosis/Resolution:

- You, or a user of your application, have specified an invalid file name. Correct the situation and try again.

errIndex (80000209H)

Raised by various collection objects if the index into the collection is invalid. Collection indexes are one-based – that is they range from 1 to the count of items in the collection.

Diagnosis/Resolution:

- This is probably a caller error.

errInvalidAddress (8000020AH)

Raised by the Header object when an attempt is made to set any of the address fields to an invalid address.

Diagnosis/Resolution:

- You, or a user of your application, have specified an invalid email address, correct the situation and retry the operation.

errPOP (8000020BH)

Raised by the POP3Client when an error occurs in the POP3 protocol.

Diagnosis/Resolution:

- Examine the contents of the LastServerResponseString property and take appropriate action before retrying the operation.

errInternalParseField (8000020CH)

Raised by the Header object when it is unable to parse the current contents of a MIME header field.

errInvalidContentType (8000020DH)

Raised by the Section object when an attempt is made to set the ContentType property to an invalid value.

Diagnosis/Resolution:

- You, or a user of your application, have specified an invalid value for the property, correct the situation and retry the operation.

errAlIreadyConnected (8000020EH)

Raised by the SMTPClient and POP3Client objects when an attempt is made to connect to an object that is already connected to a server.

Diagnosis/Resolution:

- You have attempted to call Connect or ConnectESMTP on an object that is already connected to a server. This is probably a caller error; call Close before attempting to reconnect.

errBusySending (8000020FH)

Raised by the SMTPClient and POP3Client objects when an attempt is made to perform an operation on an object that is busy. While sending and receiving messages the SMTPClient and POP3Client objects pump messages and it is possible, though undesirable, to attempt to call the object reentrantly.

Diagnosis/Resolution:

- You have attempted to make a reentrant call to an object. Guard your code with a semaphore variable.

errSocketSending (80000210H)

Raised by the SMTPClient and POP3Client objects when a socket error occurs – specifically when sending data to the server.

Diagnosis/Resolution:

- The network or server has probably failed. The error description includes a verbose explanation of the nature of the socket error that you may or may not wish to present to the user. Retry the operation at a later time.

Unused (80000211H)

This error number is currently unused.

errInvalidDisposition (80000212H)

Raised by the Section object when an attempt is made to set the Disposition property to an invalid value.

Diagnosis/Resolution:

- You, or a user of your application, have specified an invalid value for the property, correct the situation and retry the operation.

errESMTP (80000213H)

This error number is currently unused

errESMTP_AUTH (80000214H)

This error number is currently unused

errSOCKS4 (80000215H)

Raised when an error occurs while connecting through a SOCKS4 proxy.

Diagnosis/Resolution:

- You, or a user of your application, may have specified incorrect values for the connection parameters. Correct the situation and try again.

errSOCKS5 (80000216H)

Raised when an error occurs while connecting through a SOCKS5 proxy.

Diagnosis/Resolution:

- You, or a user of your application, may have specified incorrect values for the connection parameters. Correct the situation and try again.

errInvalidProxyType (80000217H)

Raised by the SMTPClient and POP3Client objects when an invalid object is passed to the SetProxyInfo method.

Diagnosis/Resolution:

- You are passing an invalid object to the SetProxyInfo method. This is probably a caller error.

Unused (80000218H-80000222)

These error numbers are currently unused

errNoCodepage (80000223H)

Raised if the Charset property of the RFC2047 object is set to character set for which there is no corresponding codepage.

Diagnosis/Resolution:

- The system does not support the character set specified. Install the operating system support for the character set / codepage or modify your code.

errNoCharset (80000224H)

Raised if the Codepage property of the RFC2047 object is set to code page for which there is no corresponding character set.

Diagnosis/Resolution:

- The system does not support the code page specified. Install the operating system support for the character set / codepage or modify your code.

Unused (80000225H-80000226)

These error numbers are currently unused

errPOP3QuickConnectStat (80000227H)

You have set optionQuickPop3Connect and called another POP3 method without issuing a POP3STAT call.

errPOP3ServerClosedPort (80000228H)

Raised if the POP3 server unexpectedly closes the connection.

Diagnosis/Resolution:

- There is a temporary or permanent error with the POP3 server. Consult with the server maintainer.

errTimeout (80000229H)

The socket operation has timed out.

Diagnosis/Resolution:

- A network or server error has caused a timeout on the connection. Try again later or diagnose network or server fault.

References

RFC 821 - Simple Mail Transfer Protocol

RFC 1725 - Post Office Protocol - Version 3

RFC 822 - Standard For The Format Of ARPA Internet Text Messages

RFC 1341 - MIME (Multipurpose Internet Mail Extensions)

RFC 2183 - Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field

License Agreement – Evaluation Edition

Codestone Ltd grants a license to use the Internet Mail Client Control Library (the library) for a period of 31 days. Copies may be made for back-up purposes only. Copies made for any other purpose are expressly prohibited.

Applications using the evaluation version of the library may not be distributed; the library must be registered before being used for any commercial purpose. The library may not be distributed by itself in any form.

This software is provided "as is". Codestone Ltd makes no warranty, expressed or implied, with regard to the software. All implied warranties, including the warranties of merchantability and fitness for a particular use, are hereby excluded.

Under no circumstances shall Codestone Ltd or the authors of this product be liable for any incidental or consequential damages, nor for any damages.

License Agreement – Developers' License Retail Edition

Codestone Ltd grants a license to use the Internet Mail Client Control Library (the library) to the original purchaser. Copies may be made for back-up purposes only. Copies made for any other purpose are expressly prohibited.

Applications using the library may be freely distributed, without royalty payments to Codestone Ltd, provided that the library is bound into these applications in such a way so as to prohibit separate use in design mode, and that the library is distributed only in conjunction with the customer's own software product. In the case of static or dynamic libraries the customer's application must included significant value added functionality and specialisation to the library. The library may not be distributed by itself in any form.

This software is provided "as is". Codestone Ltd makes no warranty, expressed or implied, with regard to the software. All implied warranties, including the warranties of merchantability and fitness for a particular use, are hereby excluded.

CODESTONE'S LIABILITY IS LIMITED TO THE PURCHASE PRICE. Under no circumstances shall Codestone Ltd or the authors of this product be liable for any incidental or consequential damages, nor for any damages in excess of the original purchase price.