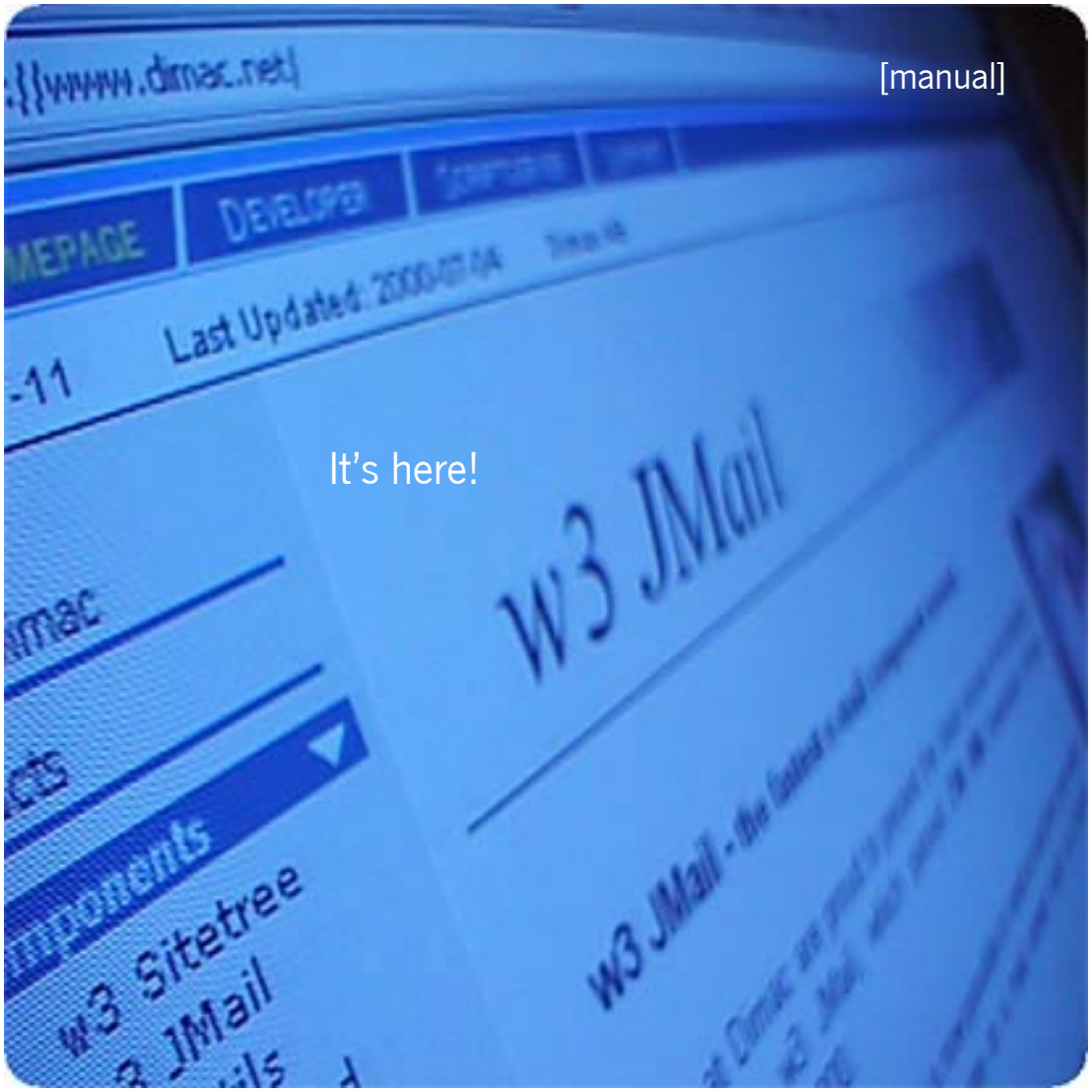
# w3 JMail 4.4

http://www.dimac.net




www.dimac.net
*websolutions*

# Contents

Manual

## The world's most popular e-mail component!

Dimac's w3 JMail is being used by 400.000+ programmers worldwide. w3 JMail is based on COM technology and can therefore be called from most modern programming languages, though it has most of its users in the ASP platform.

We are proud to present the latest version of w3 JMail, which now features sending e-mails, receiving e-mails, encryption and mass mailing with mailmerge. We believe we have made the best e-mail component even better. We have taken all the input you have given us (and that's a lot!) and most of it has been implemented in this new version, in fact we added some more stuff we thought was really cool.

What can you do with w3 JMail? A lot we say! The first thing you want to do is to add some outgoing e-mail notifications from your web pages, then move on to creating your own hotmail version using the brand new POP3 feature in w3 JMail. Securing your e-mails with PGP is a natural step into a safer communication and with the mailmerge function of w3 JMail you will be able to send newsletters to your customers without investing in expensive list servers. All in all, w3 JMail will give you new possibilities to create web applications which will rock!

Now, run the installation (which you probably already have done) and get acquainted with the world's most popular e-mail component!

Dimac Development
Duplo AB
Prästgatan 12
252 24 Helsingborg

Phone: +46 42 359400
Fax: +46 42 158050
Web site: http://www.dimac.net/
E-mail: info@dimac.net/

## What's new?

Dimac's w3 JMail 4.x features a rewritten core and truckload of new features, for example:

- The inner architecture of w3 JMail has been redesigned. E-mails are now sent using the Message Object. Instead of just one object (jmail.smtpmail) we now have more than 10 (!). For most of us jmail.message will do the trick. Note that despite all this, w3 JMail 4 is 100% backwards compatible.

- Support for receiving e-mails from POP3 mail servers.

- You can make massive bulkmailings to thousands of thousands of recipients using w3 JMails enque functions. To make it even snazzier, you can personalize each e-mail with the mailmerge object.

- Encryption of your e-mails using the most popular encryption method, PGP Encryption.

- Speedmailer, to send e-mails with just one function call.

So read on, get acquainted with the market's most popular e-mail component; Dimac's w3 JMail.

- w3 JMail comes in three different editions: Free, Standard and Professional edition. They include the following objects:

**Free Edition**
Mail.Message
JMail.Recipient
JMail.Recipients
JMail.SpeedMailer
JMail.Attachment
JMail.Headers

**Standard Edition**
(Free edition + the following)
JMail.MailMerge
JMail.PGPKeys
JMail.PGPKeyInfo
JMail.PGPDecodeResults
JMail.PGPDecodeResult
JMail.PGPDecodeResultCollection

**Professional Edition**
(Standard edition + the following)
JMail.POP3
JMail.Messages

# Installation

To use Dimac's w3 JMail you must have the jmail.dll registered at your web server. This is done by either running the installation program or by copying the jmail.dll file to your web server and manually registering it with the command regsvr32 jmail.dll. Any previous installations of w3 JMail must first be removed, as described below. To run the installation you must have administrator permissions on the web server.

Double click the w3JMail4.exe file to run the installation program. The installation will by default install at *C:\Program Files\Dimac\w3JMail4* together with this manual and some example files. More example files are viewed at our web site at http://www.dimac.net (see Products/w3 JMail). The installation will register the jmail.dll file as a COM object.

## Uninstall

Uninstall is available in the Add/Remove Programs in your settings folder. You can also do this manually by unregistering the jmail.dll component (type regsvr32 /U jmail.dll) and deleting the files.

## Enqueue and MS Pickup Directory

When using Dimac's w3 JMail as a mass mailer, you will use the method nq(). nq() on the other hand uses the SMTP service provided by Microsoft Internet Information Server (IIS).

The SMTP service has a directory where it polls for e-mails. It is called MS Pickup directory and it is by default created as C:\Inetpub\mailroot\Pickup.

1. The MS SMTP service must be installed on the machine where w3 JMail is to be used (or on a remote machine where w3 JMail can access its pickup directory). Other SMTP servers can be used as well, if they support mail delivery through a pickup directory.

2. The queueing function is dependant of w3 JMail being able to access to mail servers' pickup directory (often located in c:\inetpub\mailroot\pickup, depending on your installation). Therefore, the filesystem permissions sometimes have to be adjusted to allow the IIS guest account (IUSR) to access it.

3. Unless you are running MS Windows 2000 or later, you need to specify the location for w3 JMail to find the pickup directory. You can do this either at runtime, like this:

```
Message.MsPickupdirectory = "c:\inetpub\mailroot\pickup\"
```

or if you are using mailmerge:

```
MailMerge.BulkMerge(myRS,true,"c:\inetpub\mailroot\pickup")
```

You can also do it in your web server's registry once and for all. Read more about this in the chapter *JMail registry settings*.

4. For some installations, you will need to set read/write permissions for the EVERYONE user in the pickup directory.

*Note that only the Standard and the Professional edition of w3 JMail includes PGP Encryption!*

## Enabling PGP Encryption with w3 JMail

As Dimac's w3 JMail uses the worldwide renowned PGP to encrypt e-mails, there are some preliminary steps to take before PGP support is enabled.

First an appropriate license and installation of the PGP software has to be obtained. Useful internet links are http://www.nai.com, http://www.pgp.com and http://www.pgpi.com.

PGP and its SDK dll (pgp_sdk.dll) must be installed on the machine where w3 JMail is to be used. Version 6.5.3 of PGP were used during development.

In order to use w3 JMail and PGP from ASP pages (assuming a Windows NT/2000 system and IIS), the Internet Guest Account (most likely called IUSR _your-computer-name-here) must have at least read/write access to the PGP application and key ring files (where your encryption keys are stored). However, we recommend that you set these permissions for the EVERYONE user as well. Check PGP configuration for finding out where these files reside on your machine.

PGP settings for each user on the system are typically stored in *C:\Document and Settings\[UserName]\Application Data\PGP* on Windows 2000.

This means that the user who installed PGP, most likely the administrator, should have the PGP configuration files in his directory *C:\Documents and Settings\Administrator\Application Data\PGP*\.

The IUSR account uses the folder "Default User" and therefore the needed PGP application data has to be copied into that directory (*C:\Document and Settings\Default User\Application Data\PGP*\).

The directories *C:\Document and Settings\Default User\Application Data\PGP*\ and sometimes also *C:\Document and Settings\Administrator\Application Data\PGP*\ should have read/write permissions for the IUSR account. Replace Administrator with the user name valid in this case.

This should be pretty much the same for NT4 systems, but the folders are located in *C:\WINNT\Profiles* instead.

In most cases when encrypting e-mails with PGP using w3 JMail, the encryption key to be used is found by the e-mail addresses of the recipients. It is however possible to specify one or more encryption keys to use. A key is specified either by an e-mail address "john@hisdomain.com" or a PGP key id (ex: 0xAABBCCDD). It is possible to supply multiple keys to w3 JMail by separating the identifiers with a comma (ex: "john@hisdomain.com, 0xAABBCCDD, george@hisdomain.com" ).

**Regarding support**: please note that since PGP is NOT a Dimac product, support is not provided for problems directly related to the usage of PGP.

# Getting started

This section is divided into four parts where you will get acquainted with Dimac's w3 JMail functions for sending e-mails, receiving e-mails, encrypting e-mails and mass mailing personalized e-mails. All examples are shown in VBscript, the most common scripting language among ASP developers.

## Sending e-mails with Dimac w3 JMail

The example below shows how to get up and running with Dimac's w3 JMail. You will be shown how to create an e-mail by specifying the subject and body and how to send it.

First of all we need to create an instance of the jmail.message object:

```
set msg = Server.CreateOBject( "JMail.Message" )
```

Now lets turn on logging to make any debugging easier:

```
msg.Logging = true
```

We need to provide a sender as well as a recipient:

```
msg.From = "john.doe@mydomain.com"
msg.FromName = "John Doe"
msg.AddRecipient "lisa.simpson@springfield.com"
```

The addRecipient method can be used multiple times to add more than one recipient. Also, it can take an optional parameter which specifies the name of the recipient:

```
msg.AddRecipient "deliveryboy@futurama.com", "Fry"
msg.AddRecipient "theblob@southpark.com", "Cartman"
```

Ok, now we should add a subject:

```
msg.Subject = "How are you?"
```

and a body. The example below also shows how to add carriage returns:

```
msg.Body = "This w3 JMail stuff rocks!" & vbCrLf
```

Another way to create the body of the e-mail is to use the appendText method, which can be used multiple times to build the e-mail body:

```
msg.appendText "Here's some text."
msg.appendText "And here' s some more"
```

There you go, the e-mail message is complete, now all we need is to send it. To do that we need to enter the address of a valid mail server which accepts incoming e-mails from your web server:

```
msg.Send( "mail.myDomain.com" )
```

That's it! Once you have acquainted yourself with the basics of w3 JMail, you can find more elaborate examples at our site http://www.dimac.net/.

## Receiving e-mails with w3 JMail

Introducing with w3 JMail 4 is the ability to connect to POP3 servers and receive e-mails.

This example will receive the first e-mail in a given mailbox and display it on the web page together with its attachments which in turn are saved to the servers' disc.

First of all we have to create an instance of the JMail.POP3 object:

```
Set pop3 = Server.CreateObject( "JMail.POP3" )
```

Then we need to connect to our POP3 server, providing a user name and a password:

```
pop3.Connect "username", "password", "mail.mydomain.com"
```

Having connected to the mail server, we can now check how many e-mails reside in the mailbox:

```
Response.Write( "You have " & pop3.count &_
" e-mails in your mailbox!<br><br>" )
```

If there are any messages, we will get the first of them by using the Messages collection:

```
if pop3.count > 0 then
   Set msg = pop3.Messages.item(1)
```

Note that the Messages collection starts 1 and not 0 as most other collections and arrays do. This is because it is the standard way in the world of POP3 mail servers.

Ok, now we have an e-mail. The e-mail is an instance of the Message object that we used earlier when we sent an e-mail. This means it has all the methods and properties as the Message object has. What we want to do is to iterate through the Recipients collection that holds (naturally) all the recipients of the message and create a nicely formatted string we can use in our HTML.

```
ReTo = ""
ReCC = ""

Set Recipients = msg.Recipients
separator = ", "

For i = 0 To Recipients.Count - 1
   If i = Recipients.Count - 1 Then
      separator = ""
   End If

Set re = Recipients.item(i)
   If re.ReType = 0 Then
      ReTo = ReTo & re.Name & " "& re.EMail &""">" &_
      re.EMail & separator
   else
      ReCC = ReTo & re.Name & " "& re.EMail &""">" &_
      re.EMail & separator
   End If
Next
```

Ok, that done, let us display our e-mail message:

```
%>
<html>
   <body>
      Subject <%= msg.Subject %><br>
      From <%= msg.FromName %><br>
      Recipients To <%= ReTO %><br>
      Recipients CC <%= ReCC %>
      <hr>
      Body<br>
      <pre><%= msg.Body %></pre>
   </body>
</html>
<%
```

After that we close our POP3 connection:

```
pop3.Disconnect
```

To make it even neater we could add handling of attachments and HTML e-mails, that and much more are covered in the JMail.POP3 section later in this manual.

## Encrypting e-mails with w3 JMail

To use encryption with w3 JMail, you will need to install PGP on your web server. This is explained in detail under section *Enabling PGP Encryption with Dimac w3 JMail* at page 7.

To run this example, you must have a PGP key installed for the recipient at your web server. You can learn more about how to install PGP keys in your PGP documentation.

First start off with the code you created when you ran the first w3 JMail example. Then, right before the line that sends your e-mail:

```
msg.Send( "mail.myDomain.com" )
```

you insert the following:

```
jmail.PGPEncrypt = true
```

And that's it! w3 JMail uses PGP to encrypt your e-mails with the PGP keys installed at the web server. If the web server does not have PGP keys for the recipients, w3 JMail will throw an error.

*Note that only the Standard and the Professional edition of w3 JMail includes massmailing and mailmerge!*

## Massmailing personalized e-mails with w3 JMail

A very common way to keep in touch with your web site visitors is to send e-mails to them whenever you update your website. With w3 JMail mail-merge functions you can personalize each e-mail with the recipient's name and other details that you have collected. Mailmerge works in the way that you first create a template from which you create the e-mails. The template can contain any number of merge fields which are replaced with personal information. For example, if the following were your template:

```
Hi %%name%%!
You have %%ordersCount%% orders in process.
```

You could easily see the merge fields as they begin and end with a double percentage mark (%).

Now let us take a look at the example. First we will create a message object that will serve as a template for the merge:

```
set msgTemplate = Server.CreateObject( "JMail.Message" )

msgTemplate.Subject = "Hi %%Name%%!"

msgTemplate.Body = "Hello %%Name%%, .... bla bla..."

msgTemplate.From = "me@myDomain.com"

msgTemplate.FromName = "Mailinglist info!"

msgTemplate.AddRecipient "%%EMail%%", "%%Name%%"
```

Note how we actually use merge fields in the recipient fields as well. There, our message template is done. Next we create the mailmerge object:

```
set mMerge = Server.CreateObject( "JMail.MailMerge" )

mMerge.MailTemplate = msgTemplate
```

That was the first part, now we need a group of recipients. In this example we use an ADO recordset, however, it is possible to do it manually if you have a list of recipients on a file or so. It is assumed that we already have established a connection to our SQL server and that the connection is called sqlCon.

```
MyRS = sqlCon.execute("SELECT name, email FROM ourCustomers")
MMerge.BulkMerge myRS, false, "mail.myDomain.com"
```

As you see we enter our mail server address because the BulkMerge method sends the e-mails as soon as it has merged them, thus you don't need to use the send() method.

The BulkMerge method can also be used in conjunction with w3 JMail's enque method. Just set the second parameter to TRUE, skip the last parameter and the e-mails will be enqued, which is most often the preferred way to do it as massmailing can be a bulky operation (hence the name BulkMerge).

As BulkMerge sends e-mails as it merges, we are finished.

## Object reference

## JMail.POP3

*Note that JMail.POP3 is included only in the Professional version of w3 JMail!*

**Connect(Username, Password, Server, Port) :**

Opens the connection to the POP3 server. The Port argument is optional and defaults to 110.

```
mailbox.Connect "john", "qwerty", "mail.myDomain.com"
```

**DeleteMessages() :**

Deletes all messages from the mail server.

```
Mailbox.DeleteMessages
```

**DeleteSingleMessage(MessageID) :**

Deletes a single message on the mail server.

```
Mailbox.DeleteSingleMessage 1
```

**Disconnect() :**

Closes the connection to the server.

```
mailbox.Disconnect
```

**DownloadHeaders() :**

Downloads all headers and adds them in the Messages collection.

```
Mailbox.DownloadHeaders
```

**DownloadMessages() :**

Downloads all messages.

```
Mailbox.DownloadMessages
```

**DownloadSingleHeader(MessageID) :**

Downloads the headers of a single message, and adds them to the Messages collection.

```
Mailbox.DownloadSingleHeader 1
```

**DownloadUnreadMessages() :**

Downloads all unread (messages that have not been downloaded by ANY clientsoftware) e-mails. The e-mails are added to the messages collection. Note this command has been removed from the POP3 specification (RFC1725), so it may not be supported by all mail servers.

```
Mailbox.DownloadUnreadMessages
```

### GetLastUnreadMessage() : Integer

Returns the ID of the first unread (message that has not been downloaded by ANY client software) message. Return 0 if no messages has been accessed, -1 if this command is not supported by the server. Note this command has been removed from the POP3 specification (RFC1725), so it may not be supported by all mail servers.

```
lastMessage = Mailbox.GetLastUnreadMessage
```

### GetMessageUID(MessageID) : String

Returns the server's unique id for this message.

```
Mailbox.GetMessageUID 1
```

### Count() : Integer

Returns the number of messages on the POP3 server.

```
i = mailbox.Count
```

### Log() : String

This is the log created by w3 JMail when logging is set to TRUE.

```
Response.Write( mailbox.Log )
```

### Logging() : Boolean

Enables/Disables logging in w3 JMail. Default value is FALSE.

```
mailbox.Logging = true
```

### Messages() : Pointer

Returns the Messages object through which you can access the messages.

```
set messages = mailbox.Messages
```

### Size() : Integer

Returns the total size of your mailbox in bytes.

```
size = mailbox.Size
```

### DownloadSingleMessage(Index) : Pointer

Downloads, returns and puts the specifeid message in the Messages collection.

```
set msg = pop3.DownloadSingleMessage( 1 )
```

### Timeout : Integer

Timout in seconds for the socket used.

```
POP3.Timeout=300
```

## JMail.Messages

*Note that JMail.Messages is included only in the Professional version of w3 JMail!*

**Clear() :**

Clears the collection. Note it will NOT remove ANY e-mails from your mail server.

```
Messages.Clear
```

**Count() : Integer**

Returns the number of messages in the collection.

```
i = Messages.Count
```

**Item(Index) : Pointer**

Returns a message object.

```
set msg = Messages.Item(0)
```

## JMail.Message

#### AddAttachment(FileName, isInline, ContentType) : String

Adds a file attachment to the message. If Inline is set to TRUE, the attachment will be added as an inline attachment and addAttachment() returns the content id. This is only useful when sending HTML e-mails.

```
Message.AddAttachment("c:\autoexec.bat") or cid = Message.AddAttachment("myImage.gif", TRUE)
```

#### AddCustomAttachment(FileName, Data, isInline) : String

Adds a custom attachment. This can be used to attach “virtual files” like a generated text string or certificate etc. If Inline is set to TRUE, the attachment will be added as an inline attachment and addAttachment() returns the content id. This is only useful when sending HTML e-mails.

```
Message.AddCustomAttachment("readme.txt", "Contents of file")
```

#### AddHeader(XHeader, Value) :

Adds a user defined X-header to the message.

```
Message.AddHeader "Originating-IP","193.15.14.623"
```

#### AddNativeHeader(Header, Value) :

Adds a header to the message.

```
Message.AddNativeHeader "MTA-Settings", "route"
```

#### AddRecipient(emailAddress, recipientName, PGPKey) :

Adds a recipient to the message.

```
JMail.AddRecipient "info@dimac.net"
```

#### AddRecipientBCC(emailAddress, PGPKey) :

Adds a blind carbon copy recipient to the message. AddRecipientBCC can be used multiple times for several recipients. RecipientName is optional. PGPKey is optional, if not supplied and PGPencryption is set to TRUE, it will default to emailAddress.

```
Message.AddRecipientBCC "info@dimac.net"
```

#### AddRecipientCC(emailAddress, recipientName, PGPKey) :

Adds a carbon copy recipient to the message. AddRecipientCC can be used multiple times for several recipients. RecipientName is optional. PGPKey is optional, if not supplied and PGPencryption is set to TRUE, it will default to emailAddress.

```
Message.AddRecipientCC "info@dimac.net"
```

### AddURLAttachment(bstrURL, bstrAttachAs, isInline, bstrAuth) : String

Downloads and adds an attachment based on a URL. A seconds argument, “AttachAs”, is used for specifying the filename that the attachment will receive in the message. If Inline is set to TRUE, the attachment will be added as an inline attachment and addAttachment returns the content id. This is useful when sending HTML e-mails. A last and optional argument is used for optional WWW-Authentication.

```
Message.AddURLAttachment("http://www.mydomain.com/myfile.txt", "myfile.txt") or
Message.AddURLAttachment("http://www.mydomain.com/myfile.txt", "myfile.txt",false,
"myUserName:myPassword") or cid = Message.AddURLAttachment("http://images.dimac.net/
dimaclogo.gif", "dimaclogo.gif",true )
```

### AppendBodyFromFile(FileName) :

Clears the body of the message and replaces it with the contents of the file.

```
Message.AppendBodyFromFile "c:\mytext.txt"
```

### AppendHTML(Text) :

Append “text” to HTMLBody of message.

```
Message.AppendHTML("<h4>Hello</h4>")
```

### AppendText(Text) :

Append “text” to body.

```
JMail.AppendText "Text appended to message Body"
```

### Clear() :

Clears the message object, and gives you a new clean message.

```
Message.Clear
```

### ClearAttachments() :

Clears the list of attachments.

```
Message.ClearAttachments
```

### ClearCustomHeaders() :

Clears all custom headers.

```
Message.ClearCustomHeaders
```

### ClearRecipients() :

Clears the recipient list.

```
Message.ClearRecipients
```

### Close() :

Forces w3 JMail to close a cached connection to a mail server.

```
Message.Close();
```

### DecodeHeader(Header) : String

Decodes a message header.

```
Response.Write( Message.DecodeHeader( Headers["ReplyTo"] ) )
```

### ExtractEmailAddressesFromURL(bstrURL, bstrAuth) :

Downloads and adds e-mail addresses from a URL.

```
Message.ExtractEmailAddressesFromURL "http://www.mydomain.com/emailList.asp"
```

### GetMessageBodyFromURL(bstrURL, bstrAuth) :

Clears the body of the message and replaces it with the contents of the URL. The content type is automatically set to match the content type of the URL. The second argument (login and password) is optional.

```
Message.GetMessageBodyFromURL "http://duplo.org/", "login:password"
```

### KeyInformation(keyIdentifier) : Pointer

Returns a PGPKey object holding information for the keys matching the supplied identifier.

```
keys = Message.KeyInformation("jane@herdomain.com")
```

### LoadFromStream(Stream) :

Loads a message from a stream. Note the stream data must be compatible with the message format described in RFC822.

```
Message.LoadFromStream myStream
```

### LogCustomMessage(Message) :

Logs a custom user message to the w3 JMail log. This function works ONLY if logging is set to TRUE.

```
Message.LogCustomMessage "Hello world"
```

### nq() :

Append the e-mail to the mail queue and returns.

```
Message.nq
```

**ParseMessage(MessageSource) :**

Parses a message. MessageSource must be compatible with the message format described in RFC822.

```
Message.ParseMessage myHeaders & vbCrLf & myBody
```

**SaveToStream(Stream) :**

Saves the message source ( RFC822 compatible message ) to a stream.

```
Message.SaveToStream myStream
```

**Send(mailServer, enque) : Boolean**

Sends the message. Mail servers is a string with one or more hostnames separated by a semicolon. A username and password can also be provided for each server in the format username:password@myhost.mydomain.com.

```
Message.Send( "myMailServer" ),
Message.Send("myUserName:mypassword@mymailserver.mydomain.com")
```

**SendToNewsGroup(ServerName, Newsgroups) :**

Sends the message to Newsgroups (Separated by a ",") using the NNTP server specified.

```
SendToNewsGroup myNNTPServer, "alt.test, alt.test.test"
```

**VerifyKeys(keyString) : Boolean**

Returns TRUE if ALL the supplied keys where found in the local keyring.

```
JMail.VerifyKeys "recipient1@herdomain.com,recipient2@herdomain.com"
```

**About() : String**

Some useful information.

```
Response.Write( Message.About )
```

**Attachments() : Pointer**

Returns the Attachments collection.

```
set attachments = Message.Attachments
```

**Body() : String**

Returns the message's body.

```
Response.Write( Message.Body ) or Message.Body = "Hello world."
```

**BodyText() : String**

Returns the entire raw unparsed body ( Text - Headers.Text ).

```
Response.Write( Message.BodyText )
```

### Charset() : String

The charset of the message. The default is "US-ASCII".

```
Message.Charset = "US-ASCII"
```

### ContentTransferEncoding() : String

Specifies the content transfer encoding. The default is "Quoted-Printable".

```
Message.ContentTransferEncoding = "base64"
```

### ContentType() : String

Returns the Body's Content-Type.

```
Response.Write( Message.ContentType )
```

### Date() : Date

Returns the DateTime when the message was sent.

```
Response.Write( Message.Date )
```

### DeferredDelivery() : Date

Sets deferred delivery of messages. If the mail server supports it the message won't be delivered until this date and time.

```
Message.DeferredDelivery = CDate( "2000-12-12" )
```

### Encoding() : String

This can be used to change the default attachment encoding from base64. Valid options are "base64" or "quoted-printable".

```
Message.Encoding = "base64"
```

### EncryptAttachments() : Boolean

Set to TRUE all attachments will be encrypted too if encryption is enabled.The default value is TRUE.

```
Message.EncryptAttachments = true
```

### ErrorCode() : Integer

Contains the error code if message.silent is set to TRUE.

```
Response.Write( message.ErrorCode );
```

### ErrorMessage() : String

Contains the error message if message.silent is set to TRUE.

```
Response.Write( message.ErrorMessage );
```

### ErrorSource() : String

Contains the error source if message.silent is set to TRUE.

```
Response.Write( message.ErrorSource );
```

### From() : String

The sender's e-mail address.

```
Response.Write( Message.From ) or Message.From = "kyle@twinpeaks.com"
```

### FromName() : String

The sender's name.

```
Response.Write( Message.FromName ) or Message.FromName = "John Doe"
```

### Headers() : Pointer

Returns the header's object.

```
set Headers = Message.Headers
```

### HTMLBody() : String

Used to set and get the HTML part of the message body (if any).

```
Message.HTMLBody = "<html><body>Hello<br></body></html>"
```

### ISOEncodeHeaders() : Boolean

Encodes header strings according to iso-8859-1 character sets. The default is TRUE.

```
Message.ISOEncodeHeaders = false
```

### Log() : String

This is the log created by w3 JMail when logging is set to TRUE.

```
Response.Write( Message.Log );
```

### Logging() : Boolean

Enables/Disables logging in w3 JMail.

```
Message.Logging = true
```

### MailData() : String

The raw maildata as the e-mail will look like when it is delivered.

```
Response.write( Message.MailData )
```

### MailDomain() : String

This can be used to override the EHLO/HELO statement to your mail server.

```
Message.Maildomain = "hello.world.com"
```

### MailServerPassWord() : String

Used to specify the password for SMTP server authentication if the mail server requires a user to log in.

```
Message.MailServerPassword = "myPassword"
```

### MailServerUserName() : String

Used to specify the username for SMTP server authentication if the mail server requires a user to log in.

```
Message.MailServerUserName = "myUserName"
```

### MimeVersion() : String

Specifies the mime version. The default is "1.0".

```
Message.MimeVersion = "1.0"
```

### MsPickupdirectory() : String

The path to the pickup directory of MS SMTP service. If you run MS windows 2000, w3 JMail will autodetect the path from registry settings.

```
Message.MsPickupdirectory = "c:\myfolder"
```

### PGPEncrypt() : Boolean

Set to TRUE, the e-mail will be encrypted when the message is sent, using PGP.

```
Message.PGPEncrypt = true
```

### PGPPassphrase() : String

The PGP passphrase used when signing.

```
Message.PGPPassPhrase = true
```

### PGPSign() : Boolean

Set to TRUE, the e-mail will be signed when the message is sent, using PGP.

```
Message.PGPSign = true
```

### PGPSignkey() : String

An e-mail address or a PGP key id identifying the key to be used for signing.

```
Message.PGPSingKey = "charlie@hisdomain.com"
```

### Priority() : Byte

Returns the message' priority. 3 is normal priority.

```
Response.Write( Message.Priority ) or Message.priority = 2
```

### Recipients() : Pointer

Returns the Recipient's collection.

```
set recipients = Message.Recipients
```

### RecipientsString() : String

Readonly property of all recipients of this message.

```
Response.Write( Message.Recipients )
```

### ReplyTo() : String

Specifies an optional reply address.

```
Message.ReplyTo = "president@dimac.net"
```

### ReturnReceipt() : Boolean

Specifies whether or not the sender requires a return receipt. The default value of the property is FALSE.

```
Message.ReturnReceipt = true
```

### Silent() : Boolean

Set to TRUE, w3 JMail will not throw exceptions. Instead Message.send() will return TRUE or FALSE depending on the success of the operation.

```
Message.silent = true
```

### SimpleLayout() : Boolean

Set to TRUE to reduce the number of headers w3 JMail produces.

```
JMail.SimpleLayout = true
```

### Size() : Integer

Returns the total size of the message in bytes.

```
Response.Write( Message.Size )
```

### Subject() : String

The message's subject.

```
Response.Write( Message.Subject ) or Message.subject = "w3 JMail is here!"
```

### Text() : String

Returns the entire message source.

```
Response.Write( Message.Text )
```

### UsePipelining() : Boolean

Overrides if w3 JMail should use pipelining on a server that supports it.

```
Message.Pipelining = false
```

### Version() : String

Returns version information.

```
Response.Write( Message.Version )
```

### PGPDecode( DecodeBody, DecodeAttachments ) : Pointer

This method will decode the contents of the message object using PGP. It will decrypt the contents and verify any signatures. A passphrase must be specified for decryption. The two parameters allow for only decoding parts of a message.

Set DecodeBody to TRUE to decrypt/ verify the text parts of the message (Body and HTMLBody). Set DecodeAttachments to TRUE to decrypt/ verify the attachments of the message.

The return value is an PGPDecodeResults object, holding the results of the operation. PGP errors, signature verification status and other results for all decoded parts of the message are reported through this object.

```
decodeResults = Message.PGPDecode( true, true );
```

### EnableCharsetTranslation : boolean

Set to TRUE, which is the default, JMail will convert the text given in the Body or HTMLBody properties to the charset specified in the property Charset if needed. This is not desired if the text is already encoded. If you for instance put utf-8 encoded text into the Body , and therefore set the Charset to “utf-8”, you should set this property to FALSE.

```
Message.EnableCharsetTranslation = false;
```

### NoAutoContentID : boolean

When set to true, no Content-ID header will be set for attachments, unless they are flagged as inline (see the AddAttachment method).

## JMail.Headers

### GetHeader(Headername) : String

Returns the value of HeaderName.

```
Response.Write( Headers.GetHeader( "X-Mailer" ) )
```

### Text() : String

Returns all headers.

```
Response.Write( Headers.Text )
```

## JMail.Recipients

**Add(Value) :**

Adds a recipient to the collection.

```
Recipients.Add re
```

**Clear() :**

Clears the collection.

```
Recipients.Clear
```

**Count() : Integer**

Returns the number of recipients in the collection.

```
i = Recipients.Count
```

**Item(Index) : Pointer**

Returns a recipient object.

```
set re = Recipients.Item(0)
```

## JMail.Recipient

### New(Name, EMail, recipientType) : Pointer

Creates a new recipient, whom you can add to the Recipients collection.

```
set re = Recipient.New( "Firstname Lastname", "name@domain.com", 0 )
```

### EMail() : String

Returns the recipient's e-mail.

```
Response.Write( Recipient.EMail )
```

### Name() : String

Returns the recipient's name.

```
Response.Write( Recipient.Name )
```

### ReType() : Integer

Returns the recipient's type ( To = 0, CC = 1, BCC = 2 ).

```
Response.Write( Recipient.ReType )
```

## JMail.Attachments

**Add(Attachment) :**

Adds an attachment to the collection.

```
Attachments.Add( re )
```

**Clear() :**

Clears the collection.

```
Attachments.Clear
```

**Count() : Integer**

Returns the number of attachments in the collection.

```
i = Attachments.Count
```

**Item(Index) : Pointer**

Returns an attachment object.

```
set attachment = Attachments.Item(0)
```

## JMail.Attachment

### New(FileName, ContentType, Data) : Pointer

Creates a new attachment which you can add to the Attachments collection. If Data is specified w3 JMail creates a custom attachment containing the data, else it reads FileName from the disk.

```
set attachment = Attachment.New( "myAttachment.text", "text/plain", "this is my new
text file" )
```

### SaveToFile(FileName) :

Saves the attachment to the disk.

```
Attachment.SaveToFile "c:\incomingAttachments\" & Attachment.Name
```

### ContentType() : String

Returns the attachment's ContentType.

```
Response.Write( Attachment.ContentType )
```

### Data() : String

Returns the attachment's data.

```
Response.Write( Attachment.Data )
```

### BinaryData() : String

Returns the attachment data in binary untranslated form.

```
Response.Write( Attachment.BinaryData )
```

### isInline() : Boolean

Returns TRUE if the attachment is inline.

```
Response.Write( Attachment.IsInline )
```

### Name() : String

Returns the attachment's file name.

```
Response.Write( Attachment.Name )
```

### Size() : Integer

Returns the attachment's size.

```
Response.Write( Attachment.Size )
```

## JMail.MailMerge

*Note that JMail.MailMerge is included only in the Standard and the Professional version of w3 JMail!*

#### BulkMerge(RecordSet, enque, Maildestination) :

Merges an entire recordset with mailTemplate and sends alternatively enques it. Mail server or pickup directory is specified in mail destination.

```
MailMerge.BulkMerge myRS, false, "mail.myDomain.com"  or MailMerge.BulkMerge myRS, true, "c:\inetpub\mailroot\pickup"
```

#### Expand() : Pointer

Merges MailTemplate with user defined variables specified in the Item property.

```
MailMerge.Expand
```

#### ExpandFromRecordSet(RecordSet) : Pointer

Merges one row from an ADO Recordset with MailTemplate.

```
set msg = MailMerge.ExpandFromRecordSet( myRS )
```

#### SetDebugMode(TestMailAddress, TestCount) :

Tells Mailmerge to enter debug mode. All recipients in your e-mails will be set to TestMailAddress, TestCount e-mails will be sent to you.

```
MailMerge.SetDebugMode "myEMail@company.com", 10
```

#### Item(VariableName) : String

Sets your merging variables manually. Note you can not combine this with recordset merges.

```
MailMerge.Item( "CustomerName" ) = "Lisa Nilsson"
```

#### MailTemplate() : Pointer

Sets your own created Message object (it will serve as a template in the merge process).

```
MailMerge.MailTemplate = myMsg
```

#### MergeAttachments() : Boolean

If set to TRUE, attachments will also be scanned for merge variables and be mailmerged.

```
MailMerge.MergeAttachments = True
```

## JMail.PGPKeys

*Note that JMail.PGPKeys is included only in the Standard and the Professional version of w3 JMail!*

**Count() : Integer**

The number of keys in the collection.

```
Response.write( keys.Count )
```

**Item(Index) : Pointer**

Returns PGPKeyInfo objects from the collection.

```
key = keys.Item(0)
```

## JMail.PGPKeyInfo

*Note that JMail.MailMerge is included only in the Standard and the Professional version of w3 JMail!*

#### KeyCreationDate() : String

The date the key was created.

```
Response.write( keys.KeyCreationDate )
```

#### KeyID() : String

The id of the key.

```
Response.write( KeyID )
```

#### KeyUser() : String

The name of the user who created the key.

```
Response.write( keys.KeyUser )
```

## JMail.SpeedMailer

**EnqueMail(FromEMail, RecipientEMails, Subject, Body, MsPickupdirectory) :**

Places the e-mail in the mail queue. All data is provided through parameters.

```
SpeedMail.EnqueMail "me@mydomain.com","recipient@hisdomain.com”,"This is a test","Example"
```

**SendMail(FromEMail, RecipientEMails, Subject, Body, MailServers) :**

Sends an e-mail with SMTP, all mail data is provided through parameters.

```
SpeedMail.SendMail "me@mydomain.com","recipient@hisdomain.com","This is a test","Example","mail.mydomain.com"
```

**SendXMLMail(XML) :**

More info about ths function can be found at http://xml.dimac.net/namespace/jmail.

```
SpeedMailer.SendXMLMail XMLString
```

## JMail.PGPDecodeResults

*Note that JMail.MailMerge is included only in the Standard and the Professional version of w3 JMail!*

This object holds the individual results from a PGPDecode() operation, for each part of the message.

### Body : Pointer

This property returns a PGPDecodeResult object for the message body.

### HTMLBody : Pointer

This property returns a PGPDecodeResult object for the message html body.

### Attachments : Pointer

This property returns a PGPDecodeResultCollection object. It is a collection holding PGPDecodeResult object for each attachment in the message.

## JMail.PGPDecodeResult

*Note that JMail.MailMerge is included only in the Standard and the Professional version of w3 JMail!*

This object holds the results of a PGPDecode() operation.

**SigningUsed : boolean**

Returns true if the message part was signed, false otherwise.

**EncryptionUsed : boolean**

Returns true if the message part was encrypted, false otherwise.

**SignatureGood : boolean**

Returns true if the signature (if any) was successfully verified, false otherwise.

**Success : boolean**

Returns true if the PGPDecode() operation finished without errors, false otherwise.

**PGPErrorCode : integer**

Returns the PGPError code. If no error occured, this will be 0. Use the property “Success” to determine if the operation failed.

**PGPErrorMsg : String**

Returns a string describing the error code found in the “PGPErrorCode” property. If no error occured this will contain an empty string. Use the property “Success” to determine if the operation failed.

## JMail.PGPDecodeResultCollection

*Note that JMail.MailMerge is included only in the Standard and the Professional version of w3 JMail!*

This object is a collection holding PGPDecodeResult objects. The number of items should equal the number of attachments.

### long Count

Returns the number of items in this collection.

### PGPDecodeResul Item( index )

Returns the item with the given index. The object returned corresponds with the attachment with the same index in the Attachments collection in the message object.

# Appendix: w3 JMail Registry Settings

The registry settings are automatically set when running the installation program. You also have the possibility to set them manually.

All w3 JMail's registry keys are located in HKEY_LOCAL_MACHINE\SOFTWARE\Dimac Development\JMail

'FileAttachments'
Valid values: "true" , "false"
default-value = true

This can be used to turn off the function AddAttachment( Filename ). You can specify virtually any file on the server and attach it to an e-mail.

'AllowDownloads'
Valid values: "true" , "false"
default-value = true

Set to FALSE, the functions
- AddURLAttachment
- ExtractEmailAddressesFromURL
- GetMessageBodyFromURL
will be disabled.

'ClientLogging'
Valid values: "true" , "false"
default-value = false

If w3 JMail is used from ASP pages and this value is set to TRUE, w3 JMail will create an extra header in sent e-mails called "X-USER_IP" containing the ip address of the computer requesting the ASP page.

'POP3Enabled'
Valid values: "true", "false"
default-value = true

Set to FALSE, the POP3 support in w3 JMail will be disabled.

'NewsGroupSendEnabled'
Valid values: "true", "false"
default-value = true

Set to FALSE, the method "SendToNewsGroup" will be disabled.

'Default Mailserver'
Valid values: a string representing the name/address of the desired default mail server. If no default mail server is to be used then set the value to "" (an empty string) or remove the string value.
default-value = ""

If this field is present and not equals to "" (empty string) the name will be used as mail server regardless of whatever mail server stated with w3 JMail properties and methods.

'Default Pickupdirectory'
Valid values: a string representing the path to use as pickup directory. Leaves this field blank or deletes it if you do not wish to use a default pickup directory.
default-value = ""

If this field is present and not equals to "" (empty string) the name will be used as pickupdirectory regardless of whatever pickup directory stated in w3 JMail properties and methods.

# Software License Agreement and Limited Warranty

Notice to user:
Please read this License Carefully. This is a legal agreement between the end user ("Licensee") and Dimac Development - Duplo AB. The enclosed software and documentation are licensed by Dimac Development - Duplo AB to the original individual customer for use only on the terms described in this License Agreement (this "License"). Opening the enclosed CD envelope and/or using the Software indicates that the end user accepts and agrees to comply with these terms.

## 1. GRANT OF LICENSES

(a) Dimac Development - Duplo AB hereby grants to Licensee a non-exclusive, nontransferable, license (without the ability to sublicense) to use this product and make one copy of the Software in machine readable form for backup purposes.

(b) Dimac Development - Duplo AB retains title to the Software in all forms whatsoever.

(c) All rights not expressly granted herein are reserved by Dimac Development - Duplo AB.

## 2. LICENSE FEES

This license shall have no force or effect unless and until Licensee shall have submitted to Dimac Development - Duplo AB all applicable license fees in full. All such fees are exclusive of any taxes, duties, licenses, fees, excises or tariffs now or hereafter imposed on Licensee's production, licensing, sale, transportation, import, export or use of the Software or Licensee Programs, all of which shall be the responsibility of Licensee.

## 3. LIMITED WARRANTY

(a) Dimac Development - Duplo AB warrants that for one (1) year following delivery of the Software to Licensee, the Software, unless modified in any way by Licensee, will perform substantially the functions described in any associated product documentation provided by Dimac Development - Duplo AB. Dimac Development - Duplo AB does not warrant that the Software will meet Licensee's specific requirements or that operation of the Software will be uninterrupted or error-free. Dimac Development - Duplo AB is not responsible for any problem, including any problem which would otherwise be a breach of warranty, caused by :

* Changes in the operating characteristics of computer hardware or computer operating systems.
* Interaction of the Software with software not supplied or approved by Dimac

(b) Dimac Development - Duplo AB's entire liability and Licensee's sole remedy under the foregoing warranty during the warranty period is that Dimac Development - Duplo AB shall, at its sole and exclusive option, either use reasonable efforts to correct any reported deviation from the relevant product documentation, replace the Software with a functionally equivalent program or refund all license fees paid, in which case, this License shall immediately terminate. Any repaired or replaced Software will be rewarranted for an additional ninety (90) day period, unless subsequently modified by Licensee.

(c) The Above warranties are exclusive and no other warranties are made by Dimac Development - Duplo AB or its licensors, whether expressed or implied, including the implied warranties of merchantability, fitness for a Particular purpose, or non infringement.

## 4. LIMITATION OF LIABILITY

Under no circumstances shall Dimac Development - Duplo AB be liable for any incidental, special or consequential Damages, even if Dimac Development - Duplo AB has been advised of the possi-bility of such damages.

In no event shall Dimac Development - Duplo AB's total liability to Licensee for all damages, losses and causes of action (whether in contract, tort (including negligence) or otherwise) exceed the amount paid by Licensee for the Software.

## 5. BREACH AND TERMINATION

(a) This License is effective until terminated. This License may be terminated by the non defaulting party if either party materially fails to perform or comply with this License or any provision hereof.

(b) Termination due to a breach of Section 6 shall be effective upon notice. In all other cases termination shall be effective thirty (30) days after notice of termination to the defaulting party if the defaults have not been cured within such thirty (30) day period. The rights and remedies of the parties provided herein shall not be exclusive and are in addition to any other rights and reme-dies provided by law or this Agreement.

(c) Upon termination of this Agreement, all rights and licenses granted hereunder shall immediately terminate and all Software and other Proprietary Information of Dimac Development - Duplo AB in the possession of Licensee or under its control, shall be immediately returned to Dimac Development - Duplo AB. End user licenses properly granted pursuant to this Agreement and prior to termination of this Agreement shall not be diminished or abridged by the termination of this Agreement.

## 6. GOVERNING LAW

This Agreement is governed by Swedish law and you submit to the jurisdiction of the Swedish court in relation to any matter or dispute arising hereunder.

## 7. NOTICE TO UNITED STATES GOVERNMENT END USERS

The Software and Documentation:

* Was developed with no government funds.

* Is a trade secret of Dimac Development Duplo AB for all purposes of the Freedom of Information Act.

* Are “Commercial Items”, as that term is defined at 48 C.F.R. §2.101, consisting of “Commercial Computer Software” and “Commercial Computer Software Documentation” as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §227.7202-1 through §227.7202-4, as applicable, The “Commercial Computer Software” and “Commercial Computer Software Documentation” are being licensed to U.S. Government end users (i) only as “Commercial Items” and (ii) with only those rights granted to all other end users pursuant to the terms and conditions herein.

For units of the Department of Defense (DOD), this Software is sold only with “Restricted Rights” as that term is defined in the DOD Supplement to the Federal Acquisition Regulations (“DFARS”) 52.227-7013 (c)(1)(ii) and use, duplication or disclosure is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause of DF ARS 52.227-7013. Manufacturer: Dimac Development - Duplo AB.


Dimac Development - Duplo AB, Prästgatan 12, S-252 24 Helsingborg, Sweden