



iMSMail COM Object for inFusion Mail Server

Copyrights

inFusion Mail Server® is a product of On-Line Data Solutions, Inc.

Trademarks

inFusion Mail Server® is a registered trademark of On-Line Data Solutions, Inc.

iMS® is a registered trademark of On-Line Data Solutions, Inc.

ColdFusion is a trademark of Macromedia Corporation

BlueDragon is a trademark of New Atlanta Corporation

Suggestions and Support

Your suggestions for inFusion Mail Server and iMSMail are welcomed. Please email your comments to support@CoolFusion.com.

Technical support for iMS, excluding FusionMail and iMS-Lite, can be obtained via:

- email: support@coolfusion.com
- the iMS Support list

Details on joining the support list can be found at:
<http://www.coolfusion.com/support>.

Support for freeware products, including FusionMail and iMS-Lite, require paid support:

<http://www.coolfusion.com/commerce/index.cfm?category=6>

On-Line Data Solutions, Inc.
1919 Middle Country Road STE 204
Centereach, NY 11720-3501

Phone: (631) 737-4668

FAX: (631) 737-9539

Sales: sales@CoolFusion.com

Technical Support: support@CoolFusion.com



Copyright © 1999-2005 On-Line Data Solutions, Inc.
Manual revision 4.0

iMSMail v 4.0

iMSMail is COM automation object that can be used for easily submitting custom email jobs to iFusion Mail Server (iMS). iMSMail can be used by any program that has the ability to utilize COM objects including:

ColdFusion
PHP
ASP
Microsoft SQL Server
Microsoft Visual C++
Microsoft Visual Basic
Borland Delphi

The iMSMail DLL allows you to put email into the iMS POST queue with minimal effort. This DLL is installed into the iMS installation directory. You can move the DLL to any directory that you choose as long as you register the DLL properly using REGSVR32..

Feature overview

- Inserts mail directly into the iMS POST queue.
- Dynamic html documents with embedded content (images, etc)
- Support load balancing across multiple POST Servers.
- Supports tokenization from queries. You can therefore, send mass customized emails quickly and easily.
- Supports dynamic and fixed emails. You can create an email using the specified DLL attributes or you can specify a raw email file as well. Using raw email files allows you to take any email received by the iMS SMTP server and forward to any number of recipients (a good application for this is an email list server).
- Supports any number of email headers.
- Supports addition of multiple recipients into the same control file when the destination domain is the same. This saves on disk space and greatly improves POST processing efficiency.

- Optionally writes to a log file. This is particularly helpful when tracing log files for a particular mail. For example, a user says that they sent a mail to a particular email address but the mail never arrived. First, you look in the SMTP log file and find the record for the particular mail. Then, you look in the iMSMail log file (the log files that have file names beginning with "IMSC" and in that log file you can find the unique message ID. Finally, you check the POST log files to see the status of that message according to the message ID.
- Supports HTML and multipart mail.
- Supports RFC822-style attachments (to attach the contents of an entire email in the new email)
- Can optionally include token definitions.
- Can save a copy of the sent email to a specified directory
- Can automatically delete the raw file after sending mail.
- Support for SMTP Authentication on sent mail.
- Supports sending mail to one or more other iMS POST Servers (full UNC support)
- Exclusions parameter to remove exclusions from a recipients list.
- Qtokens are qutomatically created from query columns.
- MaxAttempts and SendingIP parameters
- Render plain text from HTML source
- Force sending of plain text or HTML exclusively
- Improved email MIME encoding

New Features in version 4.0

Version 4.0 is an almost complete rewrite from the previous version. The new version uses significantly less memory and also adds the ability to utilize ODBC datasources directly. The important new features are:

- Faster processing for large mailings
- Lower memory usage
- Automatic rollback
- New email address validator
- Rejected recipients list
- ODBC Queue support (for iMS 3.0)
- Direct ODBC queries
- Direct ODBC queries for exclusions
- Sending time designation
- Enhanced Logging
- Virtually unlimited priorities (for iMS 2.8 or higher)
- Flexible delivery retry times (for iMS 2.8 or higher)

Installation

The DLL is installed during the iMS installation process (if you option the COM Automation Object installation). If you need to register the COM object on the iMS server or any other server that will utilize the COM object then simply copy the iMSMail.dll file and register it using REGSVR32 from a command prompt. For example, if you copy the DLL to the c:\winnt\system32 folder then you would type:

```
REGSVR32 c:\winnt\system32\iMSMail.dll
```

Feature details

iMSMail, along with iMS, provides a host of advanced features and functionality that can be utilized for sending all types of emails including standard person to person emails, list emails, personalized emails (great for CRM applications) and automated emails. iMS and iMSMail are extremely flexible and fast. This section details the most important features.

Inserts mail directly into the iMS POST queue

Unlike other MAIL objects that use SMTP to send email, the iMSMail DLL inserts the email job directly into the iMS POST Server queue (the iMS POST server is the mail sending engine behind iMS) thus alleviating the additional overhead required when sending the email twice (once to get the email to the mail server and again when the mail server sends the mail to the final destination). It is also important to note that iMS is usually configured to send mail directly to its final destination so no external email server is required to set up and maintain.

Dynamic html documents with embedded content (images, etc)

When you send an email you can optionally specify images or other content that is to be embedded in the email (as opposed to linking to an image served from a remote web server). The iMSMail DLL will automatically encode and embed the content into the email and will include each item only once even if it is referenced multiple times in the HTML content.

Support load balancing across multiple POST Servers.

Email jobs can be submitted to multiple iMS POST Servers for processing. Each iMS server will receive a proportional amount of the total job. Note that you will need to configure your application, application server, web server, etc. to log in as a user with rights to the remote iMS servers. Also note that you will need to utilize the SPOOLDIR parameter and that you should use UNC path names.

Personalization via Query data.

iMS and iMSMail support the concept of Qtokens which are used to send personalized emails to recipients. Tokenization means that you can send an email with special embedded tokens (special text) that will be interpreted by the iMS server when sending mail to a specific recipient. In this way, each recipient will, for example, see their name or other personal information in the text of the email.

You can use the built-in ODBC query support of the DLL to create dynamic content. Each column from a query is treated as a Qtoken meaning that you can use any of the column information in a customized email.

For example, say you have a query like this:

```
select email, name from table1
```

In this case the name column is automatically used as a Qtoken that can be embedded into your email. So, you can then create a mail file like this:

Dear <:name:>

Thank you for your interest in our services.

The iMSMail DLL and iMS have utilized the “name:” column from the database query automatically. Please note that you must delineate the Qtoken in your email using the proper start and end tag text. The default start and end tags are “<:” and “:” respectively. However, you can use any start and end text that you wish as long as you utilize the TOKENSTART and TOKENEND parameters.

Complete header control

iMSMail supports any number of email headers. Headers are added using the SetHeader method.

For example:

```
Myobj.SetHeader("To","Our customers")
```

The example above assumes that you are creating an email from scratch (using the BODY and/or HTML attributes). If you are using an email source file (for example, you are relaying a mail to other recipients but want to customize the headers) then you can use the TOKEN_ADDHEADER and/or TOKEN_SETHEADER parameters. For example:

```
Myobj.SetToken("AddHeader_X_Test","12345");
```

Support for HTML and multipart mail

The iMSMail DLL can create plain text, HTML or multipart emails. Multipart emails are emails that contain both a plain text and HTML version of the content. An email that is in multipart format is rendered by the recipient's email program in either the plain text or HTML version depending on the end user preference. You can send multipart emails with iMSMail by specifying both the BODY and HTML attributes. You can also supply just the HTML part and use the RENDERPLAINTEXT attribute to instruct the CFX DLL to create a plain text rendering of the HTML portion.

Optional token definitions

iMSMail supports global tokens in addition to Qtokens. Global tokens are common to all recipients of an email. You may, for example, specify a token that contains the next date/time of a meeting and that will be a common item in all of the received emails. To illustrate this example:

```
<cfset nextmeeting="March 1, 2005 at 8PM">
```

```
<iMSMail Token_NextMeeting="#NextMeeting#"
body="Please be advised that our next meeting will be <:NetxMeeting:>">
```

Support for SMTP Authentication on sent mail

Although iMS can send mail directly to the recipient's email server it is also possible for iMS to relay mail through one or more local servers. If the local servers require SMTP authentication then you can use the AUTHUSER and AUTHPASS parameters to allow iMS to authenticate to the relay mail server.

Email recipient Exclusions

Sometimes it is convenient to send mass emailings by selecting a group of addresses from a database without having to worry about recipients that have specifically asked to be excluded from future mailings. In cases like these you can also specify either a list of email addresses to exclude from a mailing (EXCLUSIONS parameter) or you can specify a query for the iMSMail DLL to use when compiling a list of email addresses to exclude.

Automatic rollback

If an error occurs while sending a mailing (disk is full, for example) then all of the created files up to that point will automatically be deleted.

Email address validator

The iMSMail DLL contains an internal email address validator which is enabled by default but may be disabled by setting VALIDATEADDRESSES="NO" in the iMSMail DLL parameters. The validator will weed out addresses that are not valid and will return the list of invalid email addresses in the iMSMail_RejectedList return query.

Direct ODBC queries

The DLL has the ability to query an ODBC database directly. You can use the internal query processing either exclusively or in addition to the SMTPTO parameter which is a comma-separated list of email addresses.

Direct ODBC queries for exclusions

The DLL has the ability to query an ODBC database directly for email exclusions. Email exclusions are lists of email addresses to which the email should not be delivered. You can use the internal exclusions query processing either exclusively or in addition to the EXCLUSIONS parameter which is a comma-separated list of email addresses.

Sending time designation

You can specify a date and time for the email to be delivered (START parameter). This allows you to submit email jobs in advance that will be delivered at the designated time.

Multiple Priority Queues

The iMSMail DLL and iMS allow you to prioritize the email jobs that are submitted. A higher priority email (designated by a lower priority number) will take delivery precedence over a lower priority email. iMS version 2.8 or higher allows a virtually unlimited number of priorities (0 to 2,147,483,647) while older versions of iMS have 10 priorities (0 to 9). Priority 0 is the highest level priority.

Flexible delivery retry times (for iMS 2.8 or higher)

iMS version 2.8 or higher allows you to designate a retry time (in minutes) for every email that is sent. If the retry time is not specifically set (using the RETRYINTERVAL parameter) then the default retry time configured in the POST Server settings will be used.

When an email is delivered there may be a case where the sending needs to be retried (the recipient mailbox is full, for example). The retry timer is used to specify the number of minutes until the email delivery is reattempted.

Personalization

Personalization is one of the most important aspects of email delivery. Personalization makes possible email discussion lists, CRM applications, etc. iMS and iMSMail were designed from the ground up with flexible personalization in mind. Often, flexibility breeds abstraction. Not so with iMS. iMS allows you to easily customize every aspect of an email quickly and easily with iMSMail.

iMS and iMSMail use tokens for email personalization. Tokens are small, textual representations that are replaced with the actual values when the email is sent. For example, you may have a token called NAME that would be replaced with the actual name of the recipient.

Tokens can be global or user-specific. Global tokens will have the same value for all recipients of a particular email which user-specific tokens (Qtokens, or "Query Tokens") will be customized for each recipient. Tokens are specified in the DLL parameters with the SetToken method. For example, a global token might be designated by:

```
Myobj.SetToken("sender","sales@example.com")
```

If you then include the token "sender" in your email then the token will be replaced by the sales email address shown above.

Tokens always require a prefixed and post fixed delimiter. The default delimiters in iMS are "<:" and ":>". However, these default delimiters may be configured with different values in the iMS server configuration. If you are not sure of the setting in the iMS server then you should use the TOKENSTART and TOKENEND parameters to specify the token delimiters that you wish to use (the token delimiters should not be set to the same value).

Global tokens are of limited value when sending email. Qtokens are used much more often and allow you to send a personalized email to every recipient in a database query. When you want to send personalized mail you can use either the ODBCQUERY parameter of the DLL. The DLL will automatically utilize each column in the query for Qtokens. For example, if you have a column called "name" then there will be an available Qtoken called NAME. Please refer to the examples later on in this manual.

There are also several predefined tokens that you can use in your mailings.
These predefined tokens are as follows:

| |
|---|
| SMTPTO |
| This is the email address of the recipient |
| HEADER_XXXX |
| These are the headers that are in the email. For example, you can reference a token called HEADER_SUBJECT and, as long as the email contains a subject header, the text of the subject will replace the token in the email. |

DLL Usage

Once the DLL is installed you can utilize it in your applications as a replacement for other SMTP components. There are many advantages in using the iMSMail DLL as opposed to other available methods and, if you have iMS installed on your server, there is really no reason to ever use anything else again.

It is fairly simple to utilize the iMSMail DLL. For example, here is a simple email that is sent using ASP:

```
Set objCDOMail = Server.CreateObject("CDONTS.NewMail")
objCDOMail.From = "null@example.com "
objCDOMail.To = "joe@example.com"
objCDOMail.Subject = "Registration confirmation"
objCDOMail.Body = "You are now registered!"
objCDOMail.Send
Set objCDOMail = Nothing
```

The same email can be sent using the iMSMail DLL as follows:

```
Set objCDOMail = Server.CreateObject("iMSMail.SendMail")
objCDOMail.SMTPFrom = "null@example.com"
objCDOMail.SMTPTo = "joe@example.com"
objCDOMail.SetHeader("From", " null@example.com")
objCDOMail.SetHeader("To", " joe@example.com")
objCDOMail.SetHeader("Subject", "Registration confirmation")
objCDOMail.Body = "You are now registered!"
objCDOMail.SendMail
Set objCDOMail = Nothing
```

The biggest difference between the syntax of CDONTS and iMSMail in the above example is that the parameter count is slightly higher for the iMSMail DLL. The reason for this is that the iMSMail DLL allows you complete control over the headers of the mail and who the mail is to and from in the SMTP protocol (SMTP protocol is the language that used by mail senders and receivers). It is important to note that the headers in the mail above denoting the sender and receiver (which are displayed when you view the mail) are not necessarily the same as the sender and receiver specified when the email is transmitted. Because of this, we can set the To header="Loyal customer" and set the smtpTo="joe@example.com". In this case, joe@example.com still gets the email but the "To" address when Joe views the email will show "Loyal Customer."

iMSMail supports several methods and attributes as outlined below.

iMSMail Syntax **Details**

The following table details all of the iMSMail parameters. Although there are many available parameters you can send mails by utilizing only a few. Please refer to the examples later on in this manual.

| |
|--|
| ADDATTACHMENT(Name,ContentType) |
| The AddAttachment method is used to specify a file that is to be sent as an attachment to the email. This parameter is used only for dynamically generated emails (i.e. if the EMAILFILE parameter is not specified). Note that each entry should be the complete path to the file and that you should use double quotes around the file name if the path or file name contains spaces. The content type is optional and may be blank. |
| AUTHPASS |
| The password if authenticated SMTP is required for the receiving mail server. This parameter is only used for relaying mail through another email server that requires authentication and is normally not used if iMS is configured to send email directly to the destination. |
| AUTHUSER |
| The user name if authenticated SMTP is required for the receiving mail server. This parameter is only used for relaying mail through another email server that requires authentication and is normally not used if iMS is configured to send email directly to the destination. |
| BODY |
| The body of the email if the email is dynamically generated. The body may contain one or more tokens. |
| BOUNDARY |
| The text to use as the boundary in a MIME encoded email. The DLL creates boundaries internally if this parameter is not specified. |
| CHARSET |
| The character set of the email. The default is "iso-8859-1". Please note that the DLL does not encode the actual email – the email body should already be encoded with the character set specified. |
| CREATEMAILFILEONLY |
| Use this parameter to create the mail file only (set to true). |

| |
|--|
| DATASOURCE |
| This is the ODBC Datasource name that is to be used when querying a database for recipients and/or email exclusions (email exclusions are email addresses to which the email should not be delivered). This parameter is used in conjunction with the USERNAME and PASSWORD parameters to specify the credentials for accessing the datasource. You can also use the included ODBC credentials editor to store the username and password so that they can be omitted from your CFML scripts. |
| DELETEMAILFILE |
| If you set this parameter to true then the specified EMAILFILE will be deleted after the iMSMail DLL has been executed. |
| EMAILFILE |
| The complete path and file name of the raw email file. This parameter overrides any of the dynamic email parameters (like the body parameter and all of the header parameters). |
| ENABLEWORDWRAP |
| Turns automatic word wrapping on/off. Default is "on." |
| EXCLUSIONS |
| This is a comma-delimited list of email addresses to be removed from the list of recipients. Note that you can also use the EXODBCQUERY parameter to retrieve a list of exclusions from a database. |
| EXODBCQUERY |
| This parameter specifies an SQL query that is to be used to retrieve a list of email addresses that should not receive the mailing. The DATASOURCE parameter is required when using this parameter. |
| EXODBCQUERYFIELD |
| The name of the column that contains the email address of the excluded recipients (defaults to "email") |
| FAILTO |
| This is the address to send delivery failure notifications to. This setting overrides the SMTPFROM parameter when returning undeliverable mail. Note that the FAILTO parameter is used exclusively by the iMS server to send notifications. If you want to specify an address that the remote server should use to send bounces to then you should specify a return path using the SetHeader("Return-path","") method. |
| SETHEDER(Name,Value) |
| This method allows you to add any number of headers to a dynamically generated email. For example, to set a "TO" header which is normally required for an email, you enter the parameter like: SetHeader("To","joe@example.com") |
| HOSTNAME |
| The host name that the server sends in the helo/ehlo stage of the SMTP protocol. |

| |
|--|
| HTML |
| The body of the email in HTML format if the email is dynamically generated (i.e. the EMAILFILE parameter is not specified). |
| HTMLONLY |
| The sent email should contain only the HTML version of the email even if the BODY parameter was specified. The default is false but you can set this to true for an HTML-only email. Note that this parameter will have no effect on an email unless the HTML parameter is also specified. |
| JOBID |
| This parameter is a user-defined JOBID that will be inserted into the ODBC Queue table. This parameter applies only to IMS 3.0 or higher when using an ODBC Queue. |

| |
|--|
| LOG |
| Set this parameter to true to enable the iMSMail log file function. The default is disabled. You should also specify the LOGDIR parameters when enabling logging. |
| LOGDIR |
| The full path to the folder in which to store log files. |
| MAXATTEMPTS |
| Maximum number of sending attempts. The default is 0 which means limited only by the maximum delivery time set in the server. |
| MAXRECIPIENTS |
| The maximum number of recipients per control file. The default is 100. |
| Note: |
| The POST Server will connect to the remote mail server and deliver mail with as many recipients as possible (to save on bandwidth). If the remote server has limitations on the number of recipients then the POST Server will batch the number of recipients accordingly. So, if you have several recipients in a single control file then this is the best scenario. However, if you have several hundred or thousands of recipients in a control file, then the efficiency will decrease and it would be better to have the recipients spread out over several control files. |
| MX |
| Comma-delimited list of mail servers to send this mail to. If not specified then iMS will do an MX lookup on the recipient domain. |
| ODBCQUERY |
| This parameter specifies an SQL query that is to be used to retrieve a list of email recipients. The query must return a column of email addresses and, optionally, may contain one or more other columns to be used for personalization (Qtokens). The DATASOURCE parameter is required when using this parameter. |
| ODBCQUERYFIELD |
| The name of the column that contains the email address of the excluded recipients (defaults to "email") |
| PACKAGE |
| This true/false parameter instructs the iMSMail DLL to create packages for the iMS POST Server. Packages (available in iMS 2.5d or higher) greatly increase mail handling when there are several recipients for a particular mailing (list mail, for example). The default is true. |
| PARSETOKENS |
| Force the POST Server to parse tokens. Note: It is not necessary to specify this parameter if tokens are also specified. This parameter is supplied mainly to deal with mail files that use standard (built-in) tokens but no other tokens are specified. |

| |
|--|
| PASSWORD |
| This parameter is the password that should be used for ODBC queries. You can either specify the password and username parameters or, for added security, you can use the included ODBC Credentials editor to store the username and password instead of designating the credentials in your CFML script. |
| PLAINTEXTONLY |
| The sent email should contain only the plain text version of the email even if the HTML parameter was specified. The default is false but you can set this to true for a text-only email. Note that this parameter will have no effect on an email unless the BODY parameter is specified or the RENDERPLAINTEXT parameter has been specified along with the HTML parameter. |
| PRIORITY |
| The priority of the mail. May be set from 0 to 9 (5 is the default if not specified). 0 is the highest priority. Please note that iMS 2.8 and higher allow a virtually unlimited number of priorities (0- 2,147,483,647). An email with a higher priority (a lower numeric value) will be delivered before an email with a lower priority. |
| QUEUETYPE |
| This is the type of Queue in which to store the email job. The default is the standard iMS queue but you can optionally specify a QUEUETYPE of "ODBC" for iMS 3.0 or higher. |
| QUOTEDPRINTABLE |
| Causes iMSMail to encode the email text using quoted printable encoding. Default is false. The DLL automatically uses quoted-printable for lines that exceed the WordWrap parameter. You can turn on quoted printable by setting this parameter to true. |

| |
|--|
| RENDERPLAINTEXT |
| Instructs the DLL to create a plain text version of the HTML parameter. Note that this will cause the text in the BODY parameter to be overwritten. The default for this parameter is false but you can enable it by setting it to true |
| RETRYINTERVAL |
| The number of minutes between retries. The default is 0 which means that the server retry interval will be used. This feature is available in iMS version 2.8 or higher. |
| RFC822ATTACHMENT |
| Specifies an e-mail file to embed in the new e-mail file. You can use this parameter to send a copy of an e-mail to a recipient as an attachment. |
| SAVECOPY |
| If you set this parameter to true then a copy of the sent email will be saved in the specified "SAVEDIR" directory. |
| SAVEDIR |
| This parameter specifies the directory that the raw mail should be saved to. |
| SENDINGIP |
| The IP address to use for sending. The default is blank which means that IMS should use its default address. You can set this to be any valid IP on the server. Choosing an invalid IP will cause the server to use the Windows default address. |
| SMTPFROM |
| This is the address that the email is from in the SMTP protocol. If you omit this parameter then an address of NULL is used automatically. |
| SMTPPORT |
| The SMTP port of the receiving mail server. The default is the standard SMTP port, 25. |
| SMTPTO |
| This is a comma-delimited list of email addresses of the recipient or recipients. This field is required if you are not using the ODBCQUERY parameter. |
| SPOOLDIR |
| This parameter specifies the location of the iMS POST spool directory. iMSMail automatically determines the location of the iMS POST spool directory so you would normally not use this parameter. You can also specify a comma-delimited list of spool directories (local or on another machine) to implement load balancing. The iMSMail DLL will evenly distribute control files among the specified directories. |

START

This parameter specifies the start date and time for the email job. The default is the current date and time. This parameter must be in the following format:

yyyymmddhhnnss

where

yyyy = 4-digit year

mm = 2-digit month

dd = 2-digit day

hh = 2-digit hour (military time)

nn = 2-digit minute

ss = 2-digit second

example

If you want to post a mail to the iMS queue that is to be delivered on March 1, 2005 at 3PM then you would set Start to

20050301150000

SETTOKEN(Name,Value)

You can specify any number of token definitions for an outgoing e-mail using the SetToken method.. For example, to set a parameter called "name" you enter the token like:

SetToken("NAME","John Smith")

Please note that for email personalization it is better to use Qtokens.

TOKEN ADDHEADER

ADDHEADER is a special token that is used to add a header to an existing email. For example, you might have an email source file that is received from a list member and you want to relay the mail to all of the list recipients but you want to add an X-List header signifying the name of the list. You would use the EMAILFILE parameter to specify the email source file and then use the TOKEN ADDHEADER parameter to add you list header as follows:

```
SetToken("ADDHEADER_X_LIST","MyList")
```

Note that you can also use tokens within your header values. For example, suppose you also want to add a header designating the email address of the list member. You would add that information as follows:

```
SetToken("ADDHEADER_X_LISTMEMBER","<:SMTPTO:>")
```

There is no limit to the number of TOKEN ADDHEADER parameters.

TOKEN SETHEADER

SETHEADER is a special designation like TOKEN ADDHEADER. However, instead of adding the header, TOKEN SETHEADER will only modify a header that already exists. Suppose, for example, you want to create a threaded list archive that is based on the MESSAGE-ID of the email. You would want to remove the message-id header that was created by the email program of the list member and replace it with one of your own. In this case you would do the following:

```
SetToken("SETHEADER_MESSAGE_ID","12345")
```

Note that you can also remove existing headers from an email by setting the value to a blank string. For example, if a list member sends mail at a high priority you may want to remove that designation. You can do that as follows:

```
SetToken("setheader_X_MSMail_Priority","")
```

```
SetToken("setheader_Importance","")
```

```
SetToken("setheader_X_priority","")
```

TOKENSTART / TOKENEND

The token starting and ending delimiters (the default is "<:" and ":->"). Note: Version 1.5e and higher of iMS allows you to specify default start and end delimiters. These parameters allow you to override the default token delimiters. POST Server 1.6 is required for this functionality.

| |
|--|
| USERNAME |
| This parameter is the username that should be used for ODBC queries. You can either specify the password and username parameters or, for added security, you can use the included ODBC Credentials editor to store the username and password instead of designating the credentials in your scripts. |
| VALIDATEADDRESSES |
| This parameter tells the DLL to remove email addresses that are not in the proper format. The default is true. |
| WORDWRAP |
| Column to enforce word wrapping. Default is 70. Set to a value less than 20 to disable word wrap. This parameter is only valid for dynamically generated emails. |
| WARNTO |
| This is the address to send delivery warning notifications to. |

Return Codes and Information

iMSMail provides you with information pertaining to every aspect of the email job that was created. You can use this information to report status to users or to track the number of mails sent, number of email addresses that were rejected, etc. The following table details the return codes and information from the CFX DLL.

| |
|--|
| INPUTRECIPIENTS |
| The total number of recipients that were submitted. |
| MESSAGESIZE |
| The size of the email in bytes (this value is the same as STOREDMESSAGESIZE and is provided for backward compatibility). |
| PROCESSTIME |
| The time (in milliseconds) that the job took to run. |
| REJECTEDLISTCOUNT |
| The number of emails that were rejected due to bad formatting or being matched to the specified exclusions. |
| REJECTEDADDRESS |
| The method returns a single email that was rejected each time it is called. The method returns all of the email addresses that were rejected if the address were an invalid format or existed in the exclusions list. You can call this method REJECTEDLISTCOUNT times in order to obtain the complete list of rejected addresses. |
| RESULT |
| The result of the processing. A result of 0 is equivalent to a successful process. Any other result indicates that an error occurred during processing. |
| RESULTTEXT |
| A textual representation of the result. This value will be set to "Success" for a successful delivery and the error details if an error occurred during processing. |
| SAVEDFILENAME |
| This is the name of the file if the SAVECOPY parameter is used. This parameter will not exist if the SAVECOPY parameter was not specified. |
| STOREDFILENAME |
| This is the name of the email file that was saved to the iMS queue. |
| STOREDMESSAGESIZE |
| The size of the email in bytes. |
| TOTALEXCLUSIONS |
| The total number of recipients that were rejected due to exclusion listing. |
| TOTALRECIPIENTS |
| The total number of recipients that were added to the iMS queue. |
| IMSMAIL_VERSION |
| The version of the iMSMail DLL. |

Result Display Example

There may be times when you want to present the sender of the mail with information pertaining to the results of the CFX DLL processing. The following code is a simple example that can be expanded to suit your needs:

INCOMPLETE

DLL usage examples

The best way to learn about iMSMail is by example. This section will outline several mail sending examples from basic to advanced. When testing the DLL there are several things to keep in mind:

It is often helpful to see the control files (these are plain text INI files that contain the delivery information, tokens, recipients, etc.) and the generated email file that the CFX creates. You can designate a folder in which to create these files by using the SPOOLDIR parameter.

The examples on the following pages illustrate the basic concepts of using iMSMail.

ASP Examples

Basic Example:

In this example we're sending mail to someone who has signed up for our web site. The user has submitted a form. One of the form fields is the email address of the user so we use that information to send the email.

```
<%
```

```
' create the Mail Object
```

```
Set Mail = Server.CreateObject("iMSMail.SendMail")
```

```
dim email
```

```
email = Request.form("email")
```

```
Mail.setheader("to", email)
```

```
Mail.setheader("subject", "Thanks for registering!")
```

```
Mail.setheader("from", "admin@example.com")
```

```
Mail.body="Thanks for joining our web site! Please visit
```

```
www.example.com/confirm.asp?id=12345 in order to confirm your registration."
```

```
Mail.html="<html><body bgcolor=""blue""><font face=""arial"" color=""white"">Thanks for joining  
our web site!</font><p> Please visit <a
```

```
href=""http://www.example.com/confirm.asp?id=12345"">http://www.example.com/confirm.asp?id  
=12345</a> in order to confirm your registration.</body></html>"
```

```
' free the Mail object
```

```
Set Mail = Nothing
```

```
%>
```

Basic Example With Personalization

The first example is good for sending a quick email to a single recipient. This example will show you how to send personalized mail to one or more recipients. All of the recipient information is contained in a database and the database includes the email address as well as some personal information about each recipient. In this basic example we'll personalize the email with the email address of the recipient, their account number and their name.

```
<%  
' create mail object  
Set Mail = Server.CreateObject("iMSMail.SendMail")  
Mail.SpoolDir="h:\ims\out\  
Mail.ODBCQuery="Select email,name, account from customers"  
Mail.ODBCQueryField="email"  
Mail.Datasource="PostQueue"  
Mail.UserName="sa"  
Mail.Password="xxx"  
Mail.TokenStart="<:"  
Mail.TokenEnd=">:"  
Mail.SetHeader "To", """"<:name:>"" <:smtp:>"  
Mail.SetHeader "From", """"ACME Sales"" <sales@example.com>"  
Mail.SetHeader "X-Account", "<:account:>"  
Mail.SetHeader "Subject", "Monthly Specials"  
Mail.body="Dear <:name:>"&chr(13)&chr(10)&chr(13)&chr(10)"Please visit www.example.com  
for our monthly specials."  
Mail.sendmail()  
Response.Write"Results of mail sending:<br><br>"  
Response.Write(Mail.Result & " - " & Mail.ResultString & "<br><br>")  
Response.Write(Mail.Version&"<p>")  
Response.write("Your email was sent to ")  
Response.write(Mail.TotalRecipients)  
Response.write(" recipients from a total of ")  
'Response.write(Mail.TotalInputRecipients)  
' free the Mail object  
Set Mail = Nothing  
%>
```

This example uses the ODBCQUERY parameter instead of the SMTPTO parameter in the previous example. This allows us to send customized mail from a database query. The DLL automatically creates tokens from the query that can be used in your code (tokens are delimited markers that are replaced with actual values).

Notes

Since our query has a column called "email", we don't need to use the *ODBCQUERYFIELD* parameter although it is included in the code.

This example is using a predefined token (SMTPTO) that is always equivalent to the email address of the recipient.

All of the tokens are delimited with the default start and end token tags of "<:" and ":>". Your iMS server may be configured to use different delimiters so, to be safe, you may want to specify the token delimiters with the TOKENSTART and TOKENEND parameters.

Dynamic HTML content example

iMSMail allows you to insert embedded content into the generated html. This allows you to include images that are embedded into the rendered html instead of inserting links to content.

Example:

```
<%  
' create mail object  
Set Mail = Server.CreateObject("iMSMail.SendMail")  
Mail.SpoolDir="h:\ims\out\  
Mail.SMTPTo="jim@example.com"  
Mail.SetHeader "To", """"Jim Smith"" <:smtpo:>"  
Mail.SetHeader "From", """"ACME Sales"" <sales@example.com>"  
Mail.SetHeader "Subject", "Test image"  
Mail.html="<html><body bgcolor=blue><font face=arial  
color=white>this is a dynamic test!</font><p>Here is an  
image:<p><img src=""<imscid=h:\  
configservices.jpg|image/jpeg>""><p></body></html>"  
Mail.sendmail()  
Response.Write"Results of mail sending:<br><br>"  
Response.Write(Mail.Result & " - " & Mail.ResultString & "<br><br>")  
Response.Write(Mail.Version&"<p>")  
Response.write("Your email was sent to ")  
Response.write(Mail.TotalRecipients)  
Response.write(" recipients from a total of ")  
'Response.write(Mail.TotalInputRecipients)  
  
' free the Mail object  
Set Mail = Nothing  
%>
```

Notice that in the example above we have a new tag called "iMSCID" which is the tag that causes the DLL to insert the file as an embedded attachment. The example shows that a file called "configservices.jpg" is being inserted and that the content type is "image/jpeg."

Querying recipients and exclusions from an ODBC database

In addition to querying an ODBC datasource directly for recipients, it is also possible to query a datasource for a list of email addresses to exclude from a mailing. This is a convenient way of utilizing a list of opt-out addresses that should not receive mail. In this example we're keeping a list of opt-out addresses in a table called EXEMAILS.

```
<%  
' create mail object  
Set Mail = Server.CreateObject("iMSMail.SendMail")  
Mail.SpoolDir="h:\ims\out\  
Mail.ODBCQuery="Select email,name, account from emails"  
Mail.ExODBCQuery="Select email from exemails"  
Mail.Datasource="PostQueue"  
Mail.UserName="sa"  
Mail.Password="xxx"  
Mail.TokenStart="<:"  
Mail.TokenEnd=">"  
Mail.SetHeader "To", """"<:name:>"" <:smtpo:>"  
Mail.SetHeader "From", """"ACME Sales"" <sales@example.com>"  
Mail.SetHeader "X-Account", "<:account:>"  
Mail.SetHeader "Subject", "Monthly Specials"  
Mail.body="Dear <:name:>"&chr(13)&chr(10)&chr(13)&chr(10)&"Please visit  
www.example.com for our monthly specials."  
Mail.sendmail()  
Response.Write"Results of mail sending:<br><br>"  
Response.Write(Mail.Result & " - " & Mail.ResultString & "<br><br>")  
Response.Write(Mail.Version&"<p>")  
Response.write("Your email was sent to ")  
Response.write(Mail.TotalRecipients)  
Response.write(" recipients from a total of ")  
Response.write(Mail.InputRecipients)  
  
' free the Mail object  
Set Mail = Nothing  
%>
```

In this example we are querying from a table called "emails" in the specified datasource "postqueue." The tag will automatically produce tokens from the columns returned by the query. Any emails returned by the exclusions query will not receive the mail.

PHP Examples

Basic Example

In this example we're sending mail to someone who has signed up for our web site. The user has submitted a form. One of the form fields is the email address of the user so we use that information to send the email.

<?

```
$email = $_POST["email"];
$mail = new COM('iMSMail.SendMail');
$mail->SpoolDir = 'h:\vims\out\';
$mail->setheader('To',$email);
$mail->setheader('From','admin@example.com');
$mail->setheader('Subject','Thanks for registering!');
$mail->body='Thanks for joining our web site! Please visit
www.example.com/confirm.php?id=12345 in order to confirm your registration.';
$mail->html='<html><body bgcolor="blue"><font face="arial" color="white">Thanks for joining
our web site!<p> Please visit <a href="http://www.example.com/confirm.php?id=12345"><font
face="arial" color="white">http://www.example.com/confirm.php?id=12345</a><font face="arial"
color="white"> in order to confirm your registration.</body></html>'; $mail->sendmail();
print 'Results of mail sending:<br><br>';
print $mail->Result;
print "<br>Your email was sent to ";
print $mail->TotalRecipients;
print " recipients from a total of ";
print $mail->InputRecipients;
```

?>

Basic Example With Personalization

The first example is good for sending a quick email to a single recipient. This example will show you how to send personalized mail to one or more recipients. All of the recipient information is contained in a database and the database includes the email address as well as some personal information about each recipient. In this basic example we'll personalize the email with the email address of the recipient, their account number and their name.

```
<?
$mail = new COM('iMSMail.SendMail');
$mail->SpoolDir = 'h:\ims\out\';

$mail->ODBCQuery='Select email,name, account from customers';
$mail->ODBCQueryField='email';
$mail->Datasource='PostQueue';
$mail->UserName='sa';
$mail->Password='dev204';
$mail->TokenStart='<:';
$mail->TokenEnd='>';
$mail->SetHeader ('To', "<:name:>" "<:smtpo:>");
$mail->SetHeader ('From', "ACME Sales" "<sales@example.com>");
$mail->SetHeader ('X-Account', '<:account:>');
$mail->SetHeader ('Subject', 'Monthly Specials');
$mail->body='Dear <:name:>
```

Please visit www.example.com for our monthly specials.';

```
$mail->sendmail();
print 'Results of mail sending:<br><br>';
print $mail->Result;
print "<br>Your email was sent to ";
print $mail->TotalRecipients;
print " recipients from a total of ";
print $mail->InputRecipients;
```

```
?>
```

This example uses the ODBCQUERY parameter instead of the SMTPTO parameter in the previous example. This allows us to send customized mail from a database query. The DLL automatically creates tokens from the query that can be used in your code (tokens are delimited markers that are replaced with actual values).

Notes

Since our query has a column called "email", we don't need to use the *ODBCQUERYFIELD* parameter although it is included in the code.

This example is using a predefined token (SMTPTO) that is always equivalent to the email address of the recipient.

All of the tokens are delimited with the default start and end token tags of "<:" and ":>". Your iMS server may be configured to use different delimiters so, to be safe, you may want to specify the token delimiters with the TOKENSTART and TOKENEND parameters.

Dynamic HTML content example

iMSMail allows you to insert embedded content into the generated html. This allows you to include images that are embedded into the rendered html instead of inserting links to content.

Example:

```
<?
$email = $_POST["email"];
$mail = new COM('iMSMail.SendMail');
$mail->SpoolDir = 'h:\\ims\\out\\';
$mail->SMTPTo='jim@example.com';
$mail->SMTPFrom='null@example.com';
$mail->SetHeader ('To', "Jim Smith" <:smtpo:>);
$mail->SetHeader ('From', "ACME Sales" <sales@example.com>');
$mail->SetHeader ('Subject', 'Test image');
$mail->html='<html><body bgcolor=blue><font face=arial
color=white>this is a dynamic test!</font><p>Here is an
image:<p><p></body></html
>';
$mail->sendmail();
print 'Results of mail sending:<br><br>';
print $mail->ResultString;
print "<br>Your email was sent to ";
print $mail->TotalRecipients;
print " recipients from a total of ";
print $mail->InputRecipients;
?>
```

Notice that in the example above we have a new tag called "iMSCID" which is the tag that causes the DLL to insert the file as an embedded attachment. The example shows that a file called "configservices.jpg" is being inserted and that the content type is "image/jpeg."

Querying recipients and exclusions from an ODBC database

In addition to querying an ODBC datasource directly for recipients, it is also possible to query a datasource for a list of email addresses to exclude from a mailing. This is a convenient way of utilizing a list of opt-out addresses that should not receive mail. In this example we're keeping a list of opt-out addresses in a table called EXEMAILS.

<?

```
$mail = new COM('iMSMail.SendMail');
$mail->SpoolDir = 'h:\lms\out\';

$mail->ODBCQuery='Select email,name, account from customers';
$mail->ExODBCQuery='Select email from exemails';
$mail->ODBCQueryField='email';
$mail->Datasource='PostQueue';
$mail->UserName='sa';
$mail->Password='dev204';
$mail->TokenStart='<:';
$mail->TokenEnd='>:';
$mail->SetHeader ('To', "<:name:>" <:smtpo:>');
$mail->SetHeader ('From', "ACME Sales" <sales@example.com>');
$mail->SetHeader ('X-Account', '<:account:>');
$mail->SetHeader ('Subject', 'Monthly Specials');
$mail->body='Dear <:name:>
```

Please visit www.example.com for our monthly specials.');

```
$mail->sendmail();
print 'Results of mail sending:<br><br>';
print $mail->Result;
print "<br>Your email was sent to ";
print $mail->TotalRecipients;
print " recipients from a total of ";
print $mail->InputRecipients;
?>
```

In this example we are querying from a table called "emails" in the specified datasource "postqueue." The tag will automatically produce tokens from the columns returned by the query. Any emails returned by the exclusions query will not receive the mail.

SQL Server Stored Procedure

You can use iMSMail from within triggers or stored procedure using Microsoft SQL Server. The following is an example of a stored procedure that was written by an iMS user.

```
CREATE PROCEDURE [send_email]
```

```
@txtTo varchar(100),  
@txtFrom varchar(100),  
@txtSubject varchar(100),  
@txtBody varchar(1000)
```

```
-- Stored Procedure: Send_email  
-- Author: Tom Langer - mrtom@langerdevelopment.com  
-- This sp sends an email message using the IMSMail COM Object - it places  
-- the email message directly in the email message queue used  
-- by the inFusion Mail Server. I am using this in a real application,  
-- however, my program doesn't need so many headers/variables to be support,  
-- so I didn't create a super delux procedure here.
```

```
-- INPUTS:  
-- If you can't figure them out, email me
```

```
-- OUTPUTS:  
-- The stored procedure returns a single record recordset, containing 1  
-- field called Result, which will have only these values:  
-- The value is either FAILURE or SUCCESSFUL  
-- If successful, it may indicate whether or not there was an error or not  
-- "FAILURE - Object could not be created"  
-- "Send mail successful"  
-- "Send mail successful with Error (...)" - this result would pass what  
-- parameters encountered an error..
```

```
AS
```

```
DECLARE @object int  
DECLARE @resSP int  
DECLARE @resSM int  
DECLARE @resFN int  
DECLARE @resError varchar(8000)  
DECLARE @hr int  
DECLARE @src varchar(255), @desc varchar(255)
```

```
declare @tmpresult int
```

```

declare @tmpResultString varchar(5000)

EXEC @hr = sp_OACreate 'iMSMail.SendMail', @object OUT

-- IF THE OBJECT WAS NOT CREATED, THEN STOP THE PROCESS WITH
AN ERROR
-- MESSAGE
IF @hr <> 0
--BEGIN
-- SELECT 'FAILURE - Object could not be created ' as Result
-- RETURN
--END

BEGIN
EXEC sp_OAGetErrorInfo @object, @src OUT, @desc OUT
SELECT hr=convert(varbinary(4),@hr), Source=@src, Description=@desc
RETURN
END

SET @resFN = 0
SET @resError = ""

-- SET THE SPOOL DIRECTORY
EXEC @resSP = sp_OASetProperty @object, 'SpoolDir', 'h:\ims\out\
IF (@resSP <> 0)
BEGIN
SET @resFN = @resSP + @resFN
SET @resError = @resError + ',Error setting Spool Directory'
END

-- SET THE SMTP TO PROPERTY.. This is who the message is really sent too
EXEC @resSP = sp_OASetProperty @object, 'smtpo', @txtTo
IF (@resSP <> 0)
BEGIN
SET @resFN = @resSP + @resFN
SET @resError = @resError + ',Error setting SMTPTO'
END

-- SET THE SMTP FROM PROPERTY
EXEC @resSP = sp_OASetProperty @object, 'smtpfrom', @txtFrom
IF (@resSP <> 0)
BEGIN
SET @resFN = @resSP + @resFN
SET @resError = @resError + ',Error setting SMTPFROM'
END

```

```
EXEC @resSP = sp_OASetProperty @object, 'body', @txtBody
SET @resFN = @resSP + @resFN
```

```
SET @txtTo = 'SetHeader("To", ' + char(34) + @txtTo + char(34) + ' )'
SET @txtFrom = 'SetHeader("From", ' + char(34) + @txtFrom + char(34) + ' )'
SET @txtSubject = 'SetHeader("Subject", ' + char(34) + @txtSubject +
char(34) + ' )'
```

```
EXEC sp_OAMethod @object, @txtTo
EXEC sp_OAMethod @object, @txtFrom
EXEC sp_OAMethod @object, @txtSubject
```

```
EXEC @resSM = sp_OAMethod @object, 'SendMail'
```

```
EXEC @resSP = sp_OAGetProperty @object, 'Result', @tmpResult OUT
EXEC @resSP = sp_OAGetProperty @object, 'ResultString', @tmpResultString
OUT
```

```
-- RETURN
--print 'sending mail is complete'
```

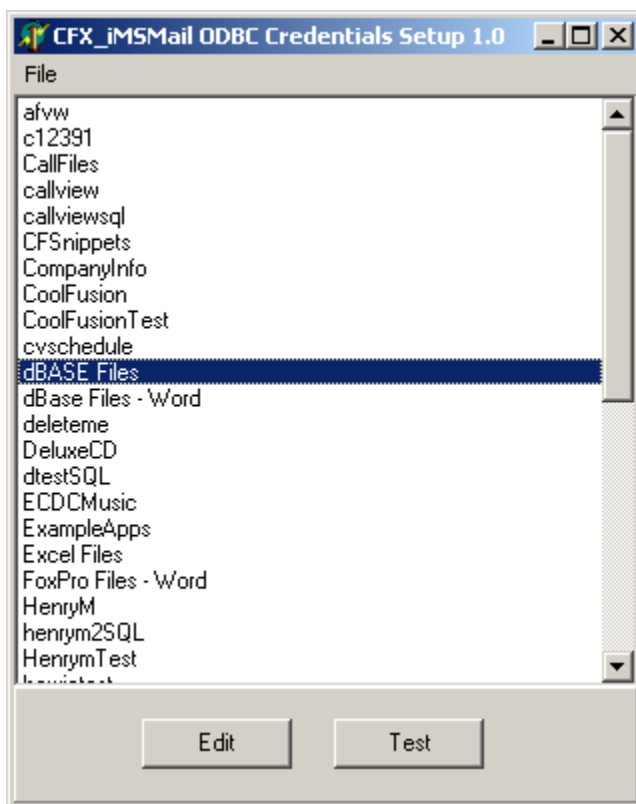
```
IF (@resSM <> 0)
BEGIN
  SELECT 'send mail process failed' as Result
  RETURN
END
```

```
IF (@resSM = 0)
BEGIN
  IF (@resFN = 0)
  BEGIN
    select 'Send mail successful' as Result
  END
  IF (@resFN <> 0)
  BEGIN
    select 'Send mail successful with Error (' + @resError + ' )' as Result
  END
END
GO
```

Preconfiguring Datasources

In order to provide additional security when directly querying datasources for email recipients or exclusions the CFX DLL has the ability to read the datasource credentials from saved values store in the system registry. The user name and password are stored in the registry using a lightweight encryption that can be quickly decoded by the iMSMail DLL.

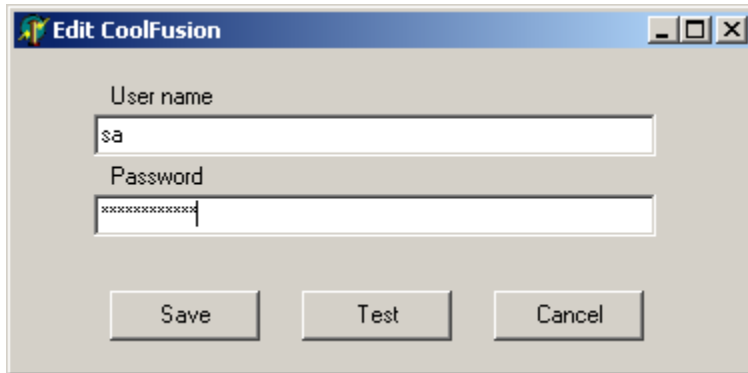
The datasource credentials are edited using the included ODBC Credentials editor. The credentials editor is accessed from the iMS program group (or the iMSMail program group if you installed the DLL separately). The credentials editor has a simple interface as shown below:



The listbox lists all of the datasources on the system. You can use the Edit or test buttons to edit the credentials or test the saved credentials assigned to the highlighted datasource.

Editing a Datasource

To edit the credentials of a datasource simply highlight the datasource and click on the Edit button.

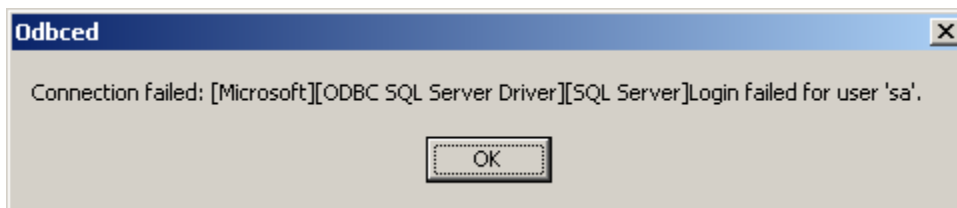


The edit box will display the existing credentials. Another test button is included on the edit box so that you can test the datasource credentials prior to saving.

If the datasource credentials are configured properly then you should see the following popup when you click on the Test button.



If the credentials are not correct then you will see something like this:



If the datasource is not configured properly then you can also see an error displayed such as:



Once you have finished editing your datasource credentials click on the Save button and the credentials will be encrypted and stored in the system registry.