MailCall™

Business Basic Email Utility
Version 2.0

http://synergetic-data.com
sdsi@synergetic-data.com

MAILCALL™ 1

BUSINESS BASIC EMAIL UTILITY 1

VERSION 2.0 1

INTRODUCTION 3

NEW IN VERSION 2 4

INSTALLATION AND SETUP 5

INSTALLATION 5

DOWNLOAD INSTALL: 5

CD INSTALL: 5

CONFIGURATION 7

LICENSING 11

CALL SYNTAX 12

SAMPLE PROGRAM 16

DIALOGS 17

INLINE ATTACHMENTS 21

Introduction

MailCall is a Business Basic program that manages the sending of Internet email from within a Business Basic program. While it has always been easy to send text email from Unix-based systems within Business Basic, it has never been possible to send binary mail with attachments without resorting to external programs or manual effort. In addition, sending email from a Windows system generally required using Windows email software, which does not lend itself to automated activities.

MailCall provides the MIME-encoded attachment capability required by many email applications, and can be integrated easily into any Business Basic application to facilitate automated email enhancements. MailCall supports both Quoted-Printable and Base64 encoding techniques, choosing which based on the file type and content.

MailCall can communicate with a SMTP server by using native tcp/ip sockets in the PRO/5 and ProvideX languages. It also can be used with older revisions of PRO/5 and BBx4, which do not have socket support, via four external programs:

- mailcall.pl, a Perl script that functions as a SMTP client in Unix-type environments
- mailcall.exe, a native Windows SMTP client executable
- Sendmail, which is often setup on Unix systems and can process MailCall submissions
- mmdf, an older Unix email system that can also process MailCall submissions

MailCall is a very small package, providing very specific functionality. It is correspondingly easy to install and use. After installation and setup, a simple CALL interface is used for all operations.

New in Version 2

MailCall Version 2 adds several new features not found in prior versions. These new features are described below:

Native socket support
Version 1 relied entirely upon external mailers for the SMTP client communication. These mailers required additional configuration that is no longer necessary. If you run in a PRO/5 or Visual PRO/5 Revision 2.2 or higher, or ProvideX, MailCall can now take advantage of native sockets, and there is no need to configure the "mailer" line in the mailcall.ini file. Note that in PRO/5 and Visual PRO/5, you must have 'alias N0 tcp' defined in your config.bbx file in order for MailCall to use native sockets. If you wish to define a different N*x* alias, specify it in stbl("$mcalias") before CALLing mailcall.bb.

When using native sockets, two new features are enabled that relate to the SMTP conversation held between MailCall and the server.

- A status window can be turned on by setting the dat.status variable to a non-zero value. This will display progress information, in character or GUI mode as appropriate, of the encoding and SMTP submission process.

- A log file can be created that provides detailed submit-response information that can be helpful when troubleshooting an emailing problem. The log file is automatically created simply be setting the dat.logfile$ variable to a log file pathname.

User Interface
There is now a user dialog provided to prompt users for information about the email. Version 1 was a pure CALL environment with no user interface. Instead, developers provided all the email information within their own programs.

By setting the new dat.dialog variable to a non-zero value, MailCall now opens an entry dialog, in character or GUI mode as appropriate, to allow the user to enter or edit email information before sending it.

Authentication
User authentication is supported (using AUTH LOGIN), which widens the support for ISP's that require authentication for their email clients. This feature is available in the new native socket feature as well as the mailcall.exe and mailcall.pl mailers. The user login information can be supplied in the mailcall.ini file or in the email data template.

Installation and Setup

MailCall consists of just a few files that can be installed in any directory on the system that will perform the email functions. That system must have TCP/IP networking setup and be able to communicate with the SMTP email server. The email server may be an in-house system, or a system hosted by an Internet Service Provider.

Installation

<u>Download Install:</u>
To install on Unix, uncompress and extract the files from mailcall.tar.Z to the mailcall directory. On Windows, use a pkzip utility, such as pkzip or winzip, to extract the files in mailcall.zip.

<u>CD Install:</u>
On Unix, first mount the CD to be readable with lowercase file names. On Unix or Windows, copy the files from the CD's /mailcall/unix or /mailcall/win directory, respectively, to the desired MailCall directory.

Once installed, you will need to modify the mailcall.ini file as specified in **Configuration**, below.

The following files are included:

addrbook.sds	Temporary internal address book file, renamed to addrbook.txt
mailcall.bb|pv	Main CALLed program (BBx4/PRO5=.bb, ProvideX=.pv)
mailcalx.bb|pv	Supplemental program
mailcall.ini	Configuration file (stores license key as well)
mailcall.exe	Windows-only SMTP client program
mailcall.pl	Unix-based SMTP client program written in perl
mailcall.pdf	This document
mailcall.sds	This is a temporary mailcall.ini file, renamed if necessary when mailcall.bb|pv is CALLed.
mc20*.bb|pv	User interface support programs
mime.typ	A standard MIME types file
readme.txt	Information not included in mailcall.pdf
license.txt	License information.

To verify the installation, start BBx or ProvideX, and try the following:

```
call “mailcall.bb”,2,x$,””              Use mailcall.pv for ProvideX
print x$
```

If the path to mailcall.bb isn’t in your prefix, you may call it with a full pathname. If you don’t see any errors, and the output from ‘print x$’ contains copyright and license information, then the install is correct and you can configure the product.

Configuration

To configure MailCall, you need to edit the mailcall.ini file, using any text editor. If you don't have a mailcall.ini file, then you can rename mailcall.sds to mailcall.ini. The following notes provide details about each option.

Native Sockets or External Mailer
If you are running ProvideX, or revision 2.2 or higher version of PRO/5 or Visual PRO/5 with 'alias N0 tcp' defined in your config.bbx file (you can specify the N*x* alias to use, if necessary, in the stbl("$mcalias")), then MailCall can use native tcp/ip sockets to communicate with the SMTP server. In this case, there is no need to configure a mailer= line.

If you *do* need an external mailer, note the following:

- On Unix, MailCall is supplied with the Perl program mailcall.pl, which is a SMTP client program designed to accept a submission file and interface with a SMTP server. In order for this program to operate, you must have a Perl interpreter. You can verify the existence of the Perl program with the Unix command 'type perl'. It should return the location of Perl as found in the system PATH variable. If your system is missing this free scripting tool, you can probably find a binary distribution on http://www.cpan.org. MailCall can also interface with sendmail or mmdf if one of those products is configured and operational.

- On Windows, we have supplied a simple Win32 executable called "mailcall.exe", which accepts the submission file and communicates with the SMTP server configured in the mailcall.ini file.

Perhaps the most important element of the configuration is to ensure the system that executes MailCall has connectivity to your SMTP mail server. This may be an in house system, or it may be hosted by your Internet Service Provider. A pretty foolproof way to test this is to telnet to port 25 on the mail server from your system (telnet *hostname* 25 from either Unix or a MS-DOS Command Window). If you get a non-error response, MailCall should work.

At version 2.0.10, the server, port, hostname, and key can be specified via the CALL template argument, meaning mailcall.ini does not need to be set up if the programmer wishes to specify these values at run-time.

server=*smtp-server*
This contains a reference to the IP address or domain name of the SMTP email server. This is used by the native socket interface, the mailcall.exe program, and the mailcall.pl program. If your mailer= setting uses sendmail or mmdf, this value is not used.

The *smtp-server* value can contain a “ssl:” prefix and a :*port* suffix, if desired, such as “ssl:smtp.gmail.com”. If the “ssl:” prefix is used, then MailCall will use SMTPS (SMTP over SSL) to communicate with the server. This provides a secure connection, and is available when PRO/5, BBj, or ProvideX support SSL socket support. Note that this protocol is not the same as TLS security, which uses a STARTTLS command to initiate the security layer after a standard connection has been established.

STARTTLS support is available starting with PxPlus 11.0 and BBj 14.10. Earlier versions of those languages do not support this method of establishing a secure connection. To use this, do not use the "ssl:" prefix, because the initial connection is not done using SSL. MailCall will automatically identify the need for STARTTLS by viewing the server headers. STARTTLS generally uses port 587, though servers can be configured to use other ports as well, such as the standard port 25, or some other port.

If STARTTLS is advertised by the mail server but there are problems with making a TLS connection, you can turn off the automatic use of it with a ‘notls:’ prefix to the server specification.

Under PRO/5 or BBj, SSL sockets are defined as distinct alias lines, using “ssl” rather than “tcp”, such as:

Alias N1 ssl

To get mailcall.bb to use an alias other than N0, you can define the t.alias$ template variable or set stbl(“$mcalias”) to the desired alias name of an SSL socket alias.

If a :*port* suffix is provided, the port number specified will be used rather than the default of 25 for standard SMTP or 465 for SMTPS.

port=*port-number*
When native sockets are used, the default SMTP port of 25 can be overridden by setting a *port-number*. Normally, this should not be required, though some ISP’s use port 587 for authenticated SMTP submissions. If the server is an “ssl:” specification, the default port is 465, rather than 25.

from=*email-address*
If no dat.from$ address is provided during the CALL to mailcall, this address is used instead.

hostname=*hostname*
If the environment does not provide a system name that is valid for the SMTP server, you can specify a value here. If no value is specified, then MailCall will determine the system hostname with the Unix “hostname” command, or on Windows with the INFO()

function in Visual PRO/5 or the NID variable in ProvideX. This element is only used by the native socket support.

login=*username*
password=*password*
If the SMTP server requires authentication, then you can define a default *username* and *password* with these elements. It is also possible to specify a *username* and *password* within the CALL interface. These values, if required, are supplied by the mail administrator, and must be supplied exactly as specified or you will probably get an authentication error and be unable to send mail.

mailer=*commandline*
If MailCall will *not* use internal sockets, then this line configures how MailCall actually sends the mail. If you are running under ProvideX or PRO/5 or Visual PRO/5 revision 2.2 or higher with a proper alias line defined, MailCall will use internal sockets and this line does not need to be configured. When required, BBx executes this command line via the SCALL() function. There must be a % character in the command line, which MailCall substitutes with the email submission file at run-time.

If no mailer value is set (all lines are commented) and a mailer is required, then a default mailer line is constructed, using “perl mailcall.pl % >mailcall.pl.log 2>mailcall.pl.err” on Unix or “mailcall.exe %” on Windows. The proper path to the mailer is automatically generated. In other words, **if you have Perl or are on Windows, there is generally no need to configure a mailer= line**.

On Windows, *commandline* should be set to the full path for mailcall.exe plus the % argument, such as ‘c:\mailcall\mailcall.exe %’. Be sure to use DOS-style backslashes rather than forward slashes.

On Unix, you will probably want to use mailcall.pl. mailcall.pl should be in the same directory as the mailcall program, and mailer should be set to the full path to mailcall.pl. The *commandline* should be ‘perl /usr/mailcall/mailcall.pl % >/dev/null’ (adjust the directory path as necessary). Perl, of course, must be installed on your system for this to work. To enable logging, change the “>/dev/null” to “>*pathname*”, and the conversation that mailcall.pl has with the SMTP server will be logged to that file.

If you use sendmail, the *commandline* ‘/usr/lib/sendmail –t <%’ should work, as it instructs sendmail to scan stdin for addresses.

If you use mmdf, then the *commandline* ‘echo $LOGNAME >%2; cat % >>%2; /usr/mmdf/bin/submit -uxto,cc* <%2; rm %2’ is used to submit email messages. The command line argument “-uxto,cc*” instructs submit to scan for To: and Cc: headers for addresses.

Note that mmdf doesn’t support Bcc: headers, while the other three methods do.

timezone=*zone*
Internet mail must include a date and time header; a properly formatted time will include your time zone. On Windows, the *zone* is added to the date and time header in the submission file. On Unix, the timezone is determined from the date command.

charset=*charsetname*
The default character set in Internet email is “us-ascii”. With this setting, it is possible to override this default for text elements of an email that includes attachments, including the body text itself.

Most configuration options have equivalent variables in the CALL string template. If you define values in the template, they override the equivalent values in the configuration file.

Licensing

MailCall is licensed by the Business Basic serial number that executes it. When first installed, MailCall operates in demonstration mode, which randomly replaces words of messages and text attachments with “*Demo” strings. This is triggered by the mailcall.ini line ‘key=demo’.

To activate MailCall, purchase an activation key for the serial number being used. It will be delivered by fax or email and is entered with a text editor into the mailcall.ini file. Modify or add the line: key=*activation-key* anywhere below the [smtp] header line. You can then test the activation by using mailcall mode 2 from Business Basic console mode, looking for the Mode: Live line.

```
READY
>call "mailcall.bb", 2, x$, ""      (use mailcall.pv on ProvideX)
>print x$
MailCall(tm)
Copyright 2000 by Allen Miglore.  All rights reserved.
Distributed under license by Synergetic Data Systems Inc.
Version: 2.0.00
Serial No: BBX468404
Key: hgfjynimhazbjq
Mode: Live
User level: multi-user
>
```

The following information is incorporated into the activation key, and will require a new key if there are any changes:

- User level – single-user or multi-user Business Basic license
- BBx or ProvideX serial number
- The first and third characters of the MailCall Version

To check the user level, you can do the following from Business Basic console mode:

BBx:	print dec(info(2,0))
Providex:	print tcb(23)

If you change these items, and your key is not “demo”, an errmsg$ value will be returned and no mail will be sent.

CALL Syntax

CALL “mailcall.bb”, mode, dat$, errmsg$

For ProvideX, use “mailcall.pv”. Note that on Windows, do not call mailcall.exe directly. The mailcall.bb and mailcall.pv programs invoke mailcall.exe when required.

Arguments:

mode is an integer value that controls how MailCall interprets or returns data in the dat$ argument. The following are valid mode values:

0	Send mail based on data in string template dat$
1	Return a string template suitable for mode=0 in dat$
2	Return version information in dat$

For modes 0 and 1, **dat$** is a string template in the format:

from:c(1*=0),to:c(1*=0),cc:c(1*=0),subject:c(1*=0),otherhead:c(1*=0),msgtxt:c(1*=0),attach:c(1*=0),status:n(1*=0),forcebase64:n(1*=0),forcenotify:n(1*=0),bcc:c(1*=0),bodymime:c(1*=0),charset:c(1*=0),timeout:n(1*=0),statuspause:n(1*=0),dialog:n(1*=0),login:c(1*=0),password:c(1*=0),logfile:c(1*=0),timezone:c(1*=0),charinterface:n(1*=0),logdata:n(1*=0)”

To provide for additions to this base template, you should always use a single CALL using mode=1, which will return a usable template in dat$.

For mode 2, dat$ returns a printable string that describes the version and license status.

Here is a description of each template field:

dat.from$ contains the sender’s email address. This value defaults to what is specified in the “from=*address*” line in mailcall.ini

**dat.to$** contains one or more email addresses delimited by commas. Note that if multiple addresses are desired, it is more common to place additional addresses in the cc$ field. Each address should be structured in one of two ways: *name@domain* or “*text name*” <*name@domain*>. It is important that if any data is present other than the plain internet email address, that the internet address be enclosed in angle brackets <>.

dat.cc$** contains zero or more carbon copy addresses. Multiple addresses must be delimited with commas. Address formats are the same as for **dat.to$, above.

dat.bcc$ contains zero or more blind carbon copy addresses. Multiple addresses must be delimited with commas. A blind carbon copy address receives a copy of the email, but the Bcc: header is removed from the submission, so no other recipients know of the Bcc: recipients.

dat.subject$ contains a single line of subject text, describing the message content.

dat.otherhead$ contains additional mail headers, should they be necessary. The rfc822 specification allows for user defined headers starting with the characters "X-", in the format of "X-*name*: *value*". Each header line should be suffixed with a CRLF (or LF) delimiter ($0D0A$). There must be no blank lines in this value, and all lines should have a proper header structure of 'name <colon (:)> <space> value'.

dat.msgtxt$ is plain text for the message body. It may contain line breaks delimited with CRLF (or LF) sequences. Lines should not exceed 900 characters without line breaks. You may also use Unix-style line break escapes (\n sequences) instead of binary CRLF characters.

**dat.bodymime$** can be used to define an alternate body text (dat.msgtxt$) MIME type. The default is "text/plain", but it is common to prepare message body text as HTML, in which case you can specify dat.bodymime$="text/html". This must be a well-known standard value (see the mime.typ file included with MailCall), and should be of the text/* family.

dat.attach$ contains one or more file names to attach to the message, delimited with commas. If this contains names, then MailCall will produce a MIME-encoded message, with the message body as plain text, text-style files (MIME types such as text/plain or text/html) as quoted-printable attachments, and other files as base64-encoded attachments.

dat.status, if set to 1 (or any positive value), will cause a status window to display as the email is processed. This flag is honored when MailCall uses native sockets or the external mailcall.exe program. When native sockets are used, the status window operates for both generation and SMTP server submission. When the external Windows mailer is used, it only operates for submission. External Unix mailers do not support this flag.

For logging on Unix installations, if you are using mailcall.pl, do this:

- Verify the setting of $log=1 in mailcall.pl near the top of the program
- Direct stdout to a file or the screen by modifying the mailer= line: something like 'perl /usr/mailcall/mailcall.pl % >/tmp/mailcall.log.' or just 'perl /usr/mailcall/mailcall.pl %'.

dat.statuspause can be set to the number of seconds to pause before closing the status window after the SMTP conversation is complete. This can help the user see the process completion without a quickly flashing window. This flag is only honored when MailCall uses native sockets and the dat.status flag is set.

dat.dialog, if set to 1, will invoke an email entry window for the user. The window is GUI or character-based, as appropriate, and provides the user with the ability to change any of the following values from the template: dat.from$, dat.to$, dat.cc$, dat.bcc$, dat.subject$,dat.attach$, dat.msgtxt$.   See the Dialogs section for more details.

dat.forcebase64, if set to 1 (or any non-zero value), will cause MailCall to always encode files with base64 encoding. By default, files whose MIME type is text are encoded using Quoted-Printable encoding.

dat.bodymime$, if set, will override the default text/plain MIME type used for the message body.

dat.charset$, if set, will override the charset default defined in the mailcall.ini configuration file, or the default of "us-ascii", when no setting is defined. Charsets are associated with any text body or attachment.

dat.login$, dat.password$, if set, and if the SMTP server requires authentication, are used for the AUTH LOGIN authentication process. These values would be provided by the ISP or mail server administrator, and must be provided exactly as specified. These values are honored when MailCall uses native sockets or the mailcall.exe or mailcall.pl mailers.

dat.logfile$, if set to a pathname, will trigger detail logging of the SMTP conversation when MailCall is using native sockets. The file will be erased and created each time MailCall is CALLed. Be careful not to use pathnames that should not be erased.

dat.timezone$, if set, will override the normal time zone value that is applied to the Date: header. The default time zone comes from either the timezone= value in mailcall.ini (for Windows) or the Unix 'date +%Z' command. Use this to set a relative GMT value, like "-0800" for PST.

dat.charinterface, if set to a non-zero value, will force character-mode for the dialog and status window displays, even in a GUI environment. The status window display affected is only the internal version used when native sockets are utilized, not the status window displayed by the mailcall.exe mailer.

dat.logdata, if set to a non-zero value, and if the dat.logfile$ is defined, and if a native socket is in use, will cause the mail submission file data to be logged to the log file specified in dat.logfile$. The default behavior is to only log SMTP conversation information, and suppress the message data.

The following template fields were added in 2.0.10:

dat.server$ names the SMTP server IP or hostname, and if specified overrides the server= setting in mailcall.ini.

dat.port, if greater than 0, overrides the port= value from mailcall.ini or the default port of 25. This value is only used when native sockets are used by mailcall.

dat.hostname$ names the local machine's hostname, and if specified overrides the hostname= setting in mailcall.ini or the value supplied by the operating system.

dat.key$ overrides the activation key specified in the key= entry in mailcall.ini.

errmsg$ will contain the text of an error message, if one occurs.

Sample Program

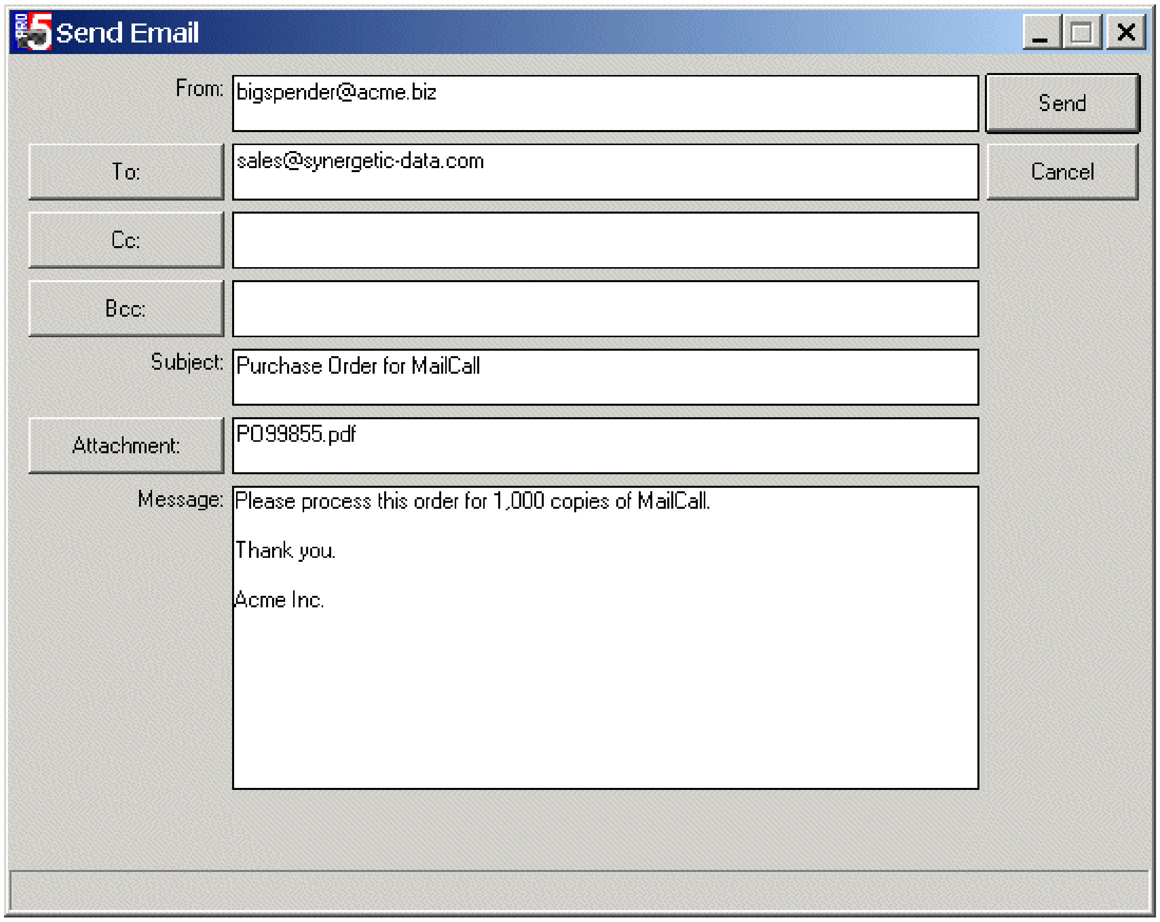
This sample program demonstrates how a program could be written to send selected customers an announcement by email.

```
init:
cust=unt
open(cust)”customers”
call “mailcall.bb”,1,t$,””
t.from$=”sales@acmecompany.com”
t.subject$=”Important Announcement from Acme Company”
t.attach$=”/tmp/announce.pdf”

nextcust:
read(cust,end=eof)*,name$,*,*,*,*,contact$,*,*,*,*,email$

if email$=”” goto nextcust
t.to$=email$

t.msgtxt$=”To “+contact$+”, “+name$+”:\n\n”
t.msgtxt$=t.msgtxt$+”Attached is an annoucement that we hope will be of
interest.\n\n”
t.msgtxt$=t.msgtxt$+”Best regards,\n\nAcme Company\n”

call “mailcall.bb”,0,t$,errmsg$
if errmsg$>”” print errmsg$,” “,; input *

goto nextcust

eof:
end
```

Dialogs

User dialogs are provided when MailCall is CALLed with the dat.dialog flag set. The window is created based upon the environment.

This window is used in Visual PRO/5.



From: bigspender@acme.biz

To: sales@synergetic-data.com

Subject: Purchase Order for MailCall

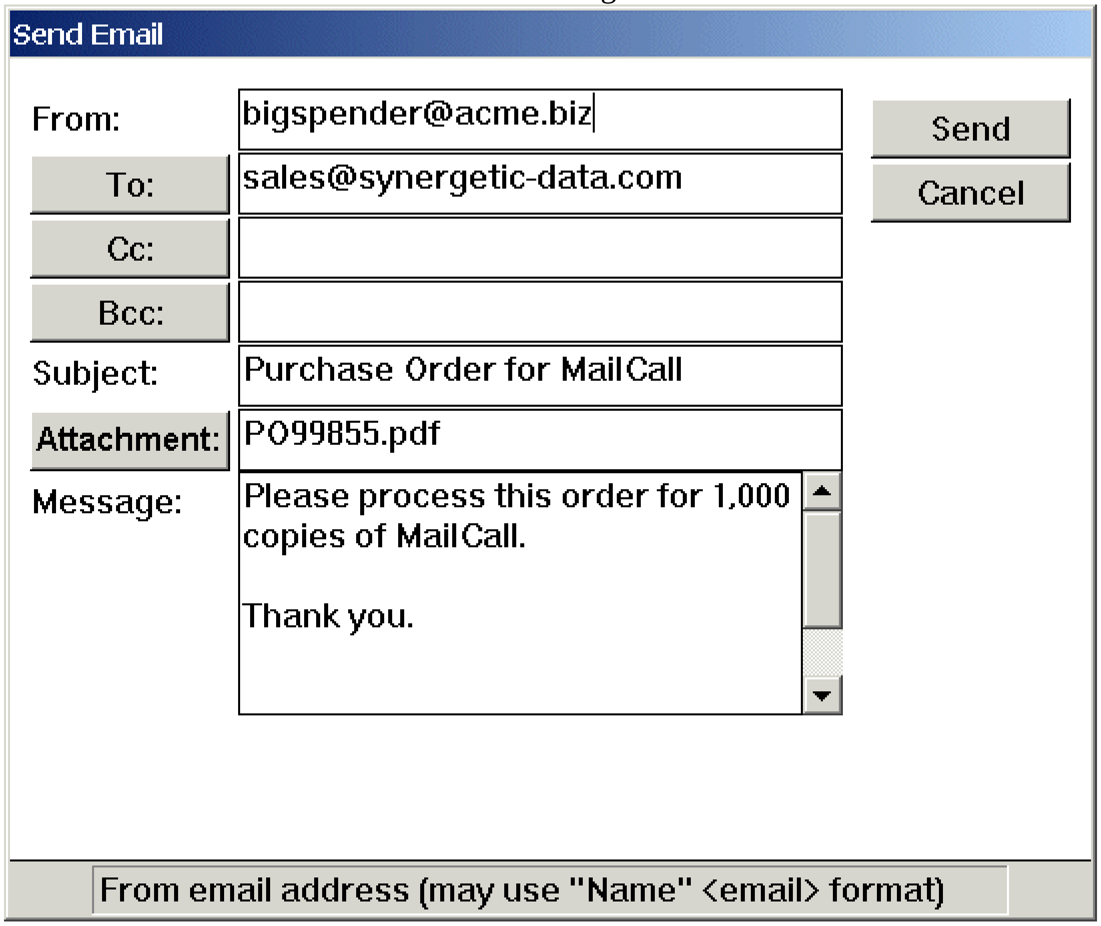
Attachment: PO99855.pdf

Message: Please process this order for 1,000 copies of MailCall.

Thank you.

Acme Inc.

This window is used in ProvideX when running under Windows or WindX.

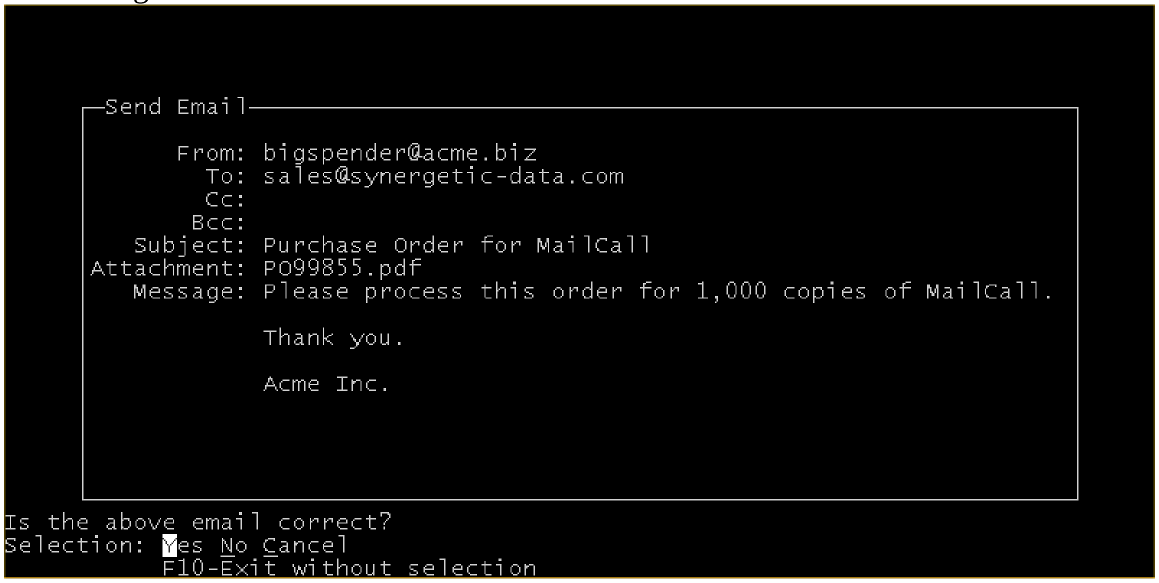


From: bigspender@acme.biz
To: sales@synergetic-data.com
Cc:
Bcc:
Subject: Purchase Order for MailCall
Attachment: PO99855.pdf
Message: Please process this order for 1,000 copies of MailCall.

Thank you.

From email address (may use "Name" <email> format)

This dialog is used for character-mode environments.



```
Send Email

      From: bigspender@acme.biz
        To: sales@synergetic-data.com
        Cc:
       Bcc:
   Subject: Purchase Order for MailCall
Attachment: PO99855.pdf
   Message: Please process this order for 1,000 copies of MailCall.

            Thank you.

            Acme Inc.

Is the above email correct?
Selection: Yes No Cancel
           F10-Exit without selection
```

The titles and prompt text of these windows is provided in the file mc20msgs.eng. The file suffix ".eng" can be overridden by setting the stbl("$mclanguage") to something other than "eng" (use gbl rather than stbl in ProvideX). This allows for translations of the MailCall interface to other languages.

Address Book

The MailCall user dialogs support a simple address book, which can be defined in the file addrbook.txt in the MailCall directory. The format of this file is very simple. It is simply a series of *name*=*email* pairs, one per line. For groups, you can delimit multiple emails with commas. To split long lists over several lines, end preliminary lines with a backslash.

Examples:

```
SDSI Sales=sales@synergetic-data.com

Bill's emails=bill@acme.com, billsmith@hotmail.com

Too Many For One Line=bill@acme.com, \
        billsmith@hotmail.com, \
        billsmith@yahoo.com
```

If you need a more comprehensive address book, then you can interface MailCall to a CALLed program which can be designed to return one or more comma-delimited email addresses. To trigger the use of your own address book program, set the stbl("$mcaddrbook") to the name of the program (use gbl rather than stbl in ProvideX).

For example, if your address book program is "abook.prg", then set stbl("$mcaddrbook","abook.prg") before CALLing MailCall with the dat.dialog flag set. When the user triggers the address book by pressing one of the To, Cc, or Bcc buttons (or pressing F2 in the appropriate field in the character-mode interface), MailCall will perform the following command:

```
Call "abook.prg",val$
```

The program abook.prg should provide a user interface to select one or more email addresses, and return them in val$. If no entry is selected, then val$ should be null. Take care to close any files and perform any other required clean up before exiting the program.

To turn off address book options altogether, set stbl("$mcaddrbook") to null (""). The address book option buttons or F2 function key will not appear in the To, Cc, and Bcc fields.

Inline Attachments

Version 2.0.15 adds support for inline attachments, where an HTML message body can refer to an attachment, such as an image. The key to this feature is the addition of a content ID value in the *t*.attach$ template variable. Each file referenced in that variable may contain an ID and tab prefix, which is removed and used as a Content-ID header for that file. The content ID can be any text value, and should be globally unique, so ideally it should contain some random data and a domain portion. The content ID can then be referred to in HTML content, using a cid:*id* syntax.

Note that some email clients may require that any attachments related to the message body be included first in the list, so it may be advisable to follow this practice.

Here is a brief example fragment:

```
Rem "invent a unique ID
Id$="logo.png."+str(tim)+"@acme.com"

Rem "attach with prefix, id + tab character + filename
t.attach$=id$+$09$+"logo.png"

t.bodymime$="text/html"

rem "refer to attachment with cid: prefix and the image's ID value
t.msgtxt$="<html><body>Here is our logo:<p>"
t.msgtxt$=t.msgtxt$+"<img src="+$22$+"cid:"+id$+$22$+">"
t.msgtxt$=t.msgtxt$+"</body></html>"
```