SMTP/POP3/IMAP Email Engine

Users Manual

(SEE_USR)

Version 8.5

August 14, 2024

This software is provided as-is.
There are no warranties, expressed or implied.

Copyright (C) 2024
All rights reserved

MarshallSoft Computing, Inc.
Huntsville AL 35815 USA

Email: info@marshallsoft.com
Web: www.marshallsoft.com

MARSHALLSOFT is a registered trademark of MarshallSoft Computing.

TABLE OF CONTENTS

TABLE OF CONTENTS - continued.

1 Introduction

The **SMTP/POP3/IMAP Email Engine Library (SEE)** is a toolkit that allows software developers to quickly develop SMTP, POP3/IMAP mail applications and can be used with any program capable of calling the Windows API.

The **SMTP/POP3/IMAP Email Engine** (SEE) is a component DLL library providing easy control of the SMTP (Simple Mail Transport Protocol), POP3 (Post Office 3), and IMAP 4 (Internet Message Access Protocol) protocols.

A simple interface provides the capability to quickly develop SMTP/POP3/IMAP email software applications to send and receive mail, including multiple MIME base64 and quoted-printable encoded attachments from within a Windows application. Knowledge of Winsock and TCP/IP is not needed.

The **SMTP/POP3/IMAP Email Engine** DLL (SEE32.DLL or SEE64.DLL) works with all versions of Windows (Windows XP through Windows 11) and can be used to write 32-bit and 64-bit applications.

The **User's Manual** applies to the **SMTP/POP3/IMAP Email Engine** (SEE) component library for all supported programming languages. It discusses SMTP POP3 IMAP email processing as well as language independent programming issues and provides purchasing and licensing information.

We have versions of the **SMTP/POP3/IMAP Email Engine SDK (SEE)** for C/C++ (SEE4C), Delphi (SEE4D), Visual Basic (SEE4VB), PowerBASIC (SEE4PB), Visual FoxPro (SEE4FP), Visual dBase (SEE4DB), Alaska Xbase++ (SEE4XB), and COBOL (SEE4CB).

Purchase a developer license for one programming language and use it with all others. All versions of the **SEE** component use the same DLLs (SEE32.DLL or SEE64.DLL). However, the examples provided for each version are written and tested for the specified programming development language. Development time is shortened because programmers need only to learn one interface.

We also have declaration files and example programs for a few other languages (such as FORTRAN and MATLAB).

Fully functional evaluation versions of our SMTP /POP3 IMAP Email software components are provided so that the developer can test the **SEE** library in their environment. The evaluation version as well as a list of the many SMTP POP3 IMAP email features provided can be found on our website at:

> http://www.marshallsoft.com/email-component-library.htm

1.1 Email Client Compatibility

The **SMTP/POP3/IMAP Email Engine** component library has been tested against multiple email clients, including Eudora, Microsoft Outlook, Outlook Express, Thunderbird, Pegasus, Calypso, PM Mail 98, Actif Mail, Lotus Notes, and Netscape.

The library has also been tested against a variety of UNIX and Windows servers on our LAN and on the Internet.

1.2 Documentation Set

The complete set of documentation is divided into three manuals in Adobe PDF format. This is the second manual (SEE_USR.PDF) in the set.

- SEE_4x Programmer's Manual `(SEE_4x.PDF)`
- SEE User's Manual `(SEE_USR.PDF)`
- SEE Function Reference Manual `(SEE_REF.PDF)`

The "x" in SEE_4x Programmer's Manual specifies the host programming language such as C for C/C++, D for Delphi, VB for Visual Basic, PB for Power BASIC, FP for FoxPro, DB for dBase, and XB for Xbase.

The **SMTP/POP3/IMAP Programmer's Manual** (**SEE_4x.PDF**) is the programming language specific manual. All language dependent programming issues such as compiling, compilers and example programs are discussed here.

The **SMTP/POP3/IMAP User's Manual (SEE_USR.PDF**) discusses language independent SMTP/POP3 email processing issues. License and purchase information is also provided.

The **SMTP/POP3/IMAP Reference Manual** (**SEE_REF.PDF**) contains specific details on each individual **SEE** email function. It also includes an error return code list.

1.3 Technical Support

We want you to be successful in developing software applications using our **SMTP/POP3/IMAP Email Library**! We are committed to providing the best, most robust software development toolkit that we can. If you have any suggestions for enhancements or comments, please let us know.

If you are having a problem using **SEE**, refer to section 10.0 "Resolving Problems".

Registered users with a current license (purchase or last update) within the last 12 months, can email us at info@marshallsoft.com.

To avoid having your email deleted by our SPAM scanners, begin the subject with "MSC HELP" or with the product name (SEE4C, SEE4VB, etc.). Zip up any attachments and send plain ASCII text email only.

Contact us by email at info@marshallsoft.com

The latest versions of our products are available on our web site at

> http://www.marshallsoft.com

and on our anonymous FTP site at

> ftp://ftp.drivehq.com/MarshallSoft/PublicFolder/

Registered users with a current license (12 months or less) can update to the latest DLL's at

> http://www.marshallsoft.com/oem.htm

1.4 How to Purchase

A developer license for the professional version of the **SMTP/POP3/IMAP Email Library SDK** (SEE) may be purchased for $119 (US dollars) for electronic delivery. This price is good for one year from the release date. The professional version DLL is also branded internally with your company name or Customer ID.

The fastest and easiest way to order is on our web site at

      https://www.marshallsoft.com/vmorder.htm

You can also order by completing INVOICE.TXT (Pro Forma invoice) and emailing (info [at] marshallsoft.com), mailing (see our address at top), or faxing it to us. Our fax number will be provided upon request.

Multiple copy discounts (3 or more) and site licenses are available. Please call for details.

We accept American Express, VISA, MasterCard, Discover, checks in US dollars drawn on a US bank, International Postal Money Orders, Pay Pal and Western Union payments.

The registered package includes:
(1) Win32 and Win64 DLL Libraries without the evaluation version popup window.
(2) Email support for one year.
(3) Free downloadable updates to the registered DLLs for one year.

Note that the SEE DLLs never expire.

1.5 Academic Discount

We offer an "academic price" with a 40% discount for prepaid email orders to faculty and <u>full</u> time students currently enrolled in any accredited high school, college, or university. The software must be used for educational purposes.

To qualify for the discount, your school must have a web site and you must have a current email address (not forwarded) at your school. On the online order form on our web site order page, put "academic discount", or enter "student at" (or "faculty at") and your schools web site address (URL) in the comments. Your order will be sent to your email address at your school.

This offer is not retroactive and cannot be used with any other discount. Products bought with academic pricing cannot be used for any commercial purpose.

1.6 Updates

When a developer license is purchased for the **SMTP/POP3/IMAP Email Library SDK**, the developer will be sent a new registered DLL plus a license file (SEExxxx.LIC) that can be used to update the registered DLL for a period of <u>one year</u> from purchase. Updates can be downloaded from

http://www.marshallsoft.com/update.htm

After one year, the developer license must be updated to be able to download updates. The developer license can be updated for $33 if ordered within <u>one year</u> from the original purchase (or previous update). After one year, licenses can be updated for $55 ($77 after 3 years).

Note that the registered SEE DLL never expires.

Also see file UPDATES.TXT.

1.7 Customer ID

The Customer ID is a 4-6-digit number following the product name (SEE) in the license file. For example, customer 12345 will receive license file **SEE12345.LIC**. Provide the Customer ID in the SUBJECT of an email when contacting us for technical support (SEE4C 12345).

1.8 License File

A license file SEExxxx.LIC, where "xxxxx" is the 4-6-digit customer ID, is provided with each developer license. The license file is an encrypted binary file used for updating SEE as explained in section 1.6 "Updates". The license file is required in order to create (or update) the registered DLL's.   The license file can be found in the /DLLS directory created after SETUP is run.

1.9 Distribution

To run an application (that calls SEE functions) on another computer, the file SEE32.DLL (or SEE64.DLL) must be copied to the Windows directory of the other computer. The Windows directory is normally C:\WINDOWS. Do not attempt to "register" the DLLs.

2 SMTP/POP3/IMAP Library Overview

2.1 Dynamic Link Libraries

The **SMTP/POP3/IMAP Email Library SDK** consists of a Win32 [SEE32.DLL] and a Win64 [SEE64.DLL] dynamic link library (DLL). A DLL is characterized by the fact that it need not be loaded until required by an application program and that only one copy of the DLL is necessary regardless of the number of application programs that use it. Contrast this to a static library that is bound at link time to each and every application that uses it.

2.2 Keycode

When a developer license is purchased, the developer will receive a new SEE32.DLL and SEE64.DLL and a keycode for the DLLs. Pass this keycode as the argument to **seeAttach**. The keycode will be found in the file named "KEYCODE". The keycode for the evaluation version is 0. The keycode for the registered version will be a unique 9 or 10 digit number. Note: Your keycode is NOT your Customer ID/Registration number.

2.3 GUI and Console Mode

SMTP/POP3/IMAP Email Library (SEE) functions can be called from WIN32 console mode programs as well as GUI programs. A "console mode" program is a Windows 95/98/ME/2003/NT/2000/XP/Vista/Win7/Win8 WIN32/Win64 command line program running in a command window. Although console mode programs look like DOS programs, they are WIN32/Win64 programs that have access to the entire Windows address space.

2.4 Getting Started Using the Library

The first **SMTP/POP3/IMAP (SEE)** function that should be called is **seeAttach**, which initializes the **SEE** library and allocates necessary resources. **seeAttach** is typically called in the initialization section of the application and should be called just once.

After **seeAttach** is called, you are ready to connect to an SMTP server with **seeSmtpConnect** or a POP3/IMAP server with **seePop3Connect**. Once connected you are ready to call the other SMTP or POP3 functions in order to send or receive email.

After sending or receiving email, the connection to the server can be closed with **seeClose**. **seeClose** should not be called if the previous **seeSmtpConnect** or **seePop3Connect** failed.

Before exiting the application, **seeRelease** should be called. **seeRelease** should not be called if **seeAttach** failed.

The best way to get familiar with **SEE** is to try out one of the example programs. The example programs are described in the SEE_4x Programmer's Manual. The "x" in SEE_4x specifies the host language such as C for C/C++, VB for Visual Basic, etc. the example source is written in.

```
• SEE_4C    SMTP/POP3 (SEE) Programmer's Manual for C/C++
• SEE_4D    SMTP/POP3 (SEE) Programmer's Manual for Delphi
• SEE_4VB   SMTP/POP3 (SEE) Programmer's Manual for Visual Basic
• SEE_4PB   SMTP/POP3 (SEE) Programmer's Manual for PowerBASIC
• SEE_4FP   SMTP/POP3 (SEE) Programmer's Manual for Visual FoxPro
• SEE_4DB   SMTP/POP3 (SEE) Programmer's Manual for Visual dBase
• SEE_4XB   SMTP/POP3 (SEE) Programmer's Manual for Xbase++
• SEE_4CB   SMTP/POP3 (SEE) Programmer's Manual for COBOL
```

2.5 Application Program Logic

Application programs developed using the **SMTP/POP3/IMAP Email** library should follow this logic:

SENDING MAIL:

```
seeAttach
  seeSmtpConnect
  LOOP
    Any SMTP command such as seeSendEmail
  END-LOOP
  seeClose
seeRelease
```

CHECKING MAIL:

```
seeAttach
  seePop3Connect (or seeImapConnect)
  LOOP
    Any POP3 (or IMAP) command such as seeGetEmailFile
  END-LOOP
  seeClose
seeRelease
```

CHECK MAIL THEN SEND MAIL

```
seeAttach
  seePop3Connect (or seeImapConnect)
  LOOP
    Any POP3 (or IMAP) command such as seeGetEmailFile
  END-LOOP
  seeClose
  seeSmtpConnect
  LOOP
    Any SMTP command such as seeSendEmail
  END-LOOP
  seeClose
seeRelease
```

3 Windows Search Path

When installing SEE, the DLL's (SEE32.DLL and SEE64.DLL) are normally copied to the C:\WINDOWS directory, which is in the Windows search path. The DLL's can optionally be moved anywhere in the Windows Search Path.

The Windows Search Path can be edited as necessary. Enter "System Variables Path" in Windows search box. When the "System Properties" window is displayed, choose "Environment Variables" then edit the "PATH" system variable.

4 Email Basics

An email account is normally hosted on a computer that has a permanent connection to the Internet. Outgoing email is sent over the Internet using the SMTP (Simple Mail Transport Protocol) and is stored in mass storage until retrieved using the POP3 (Post Office Protocol 3) or IMAP (Internet Message Access Protocol) protocol. POP3 is a subset of IMAP, so that the POP3 protocol can be used to download email from an IMAP server.

The **SMTP/POP3/IMAP Email Engine (SEE)** component library is an "email client" and is used to send and receive email using the SMTP and POP3 protocols.

4.1 SMTP Servers

A connection to a SMTP server must be made to be able to send email. There are several common configurations for SMTP servers. Connect to SMTP servers with **seeSmtpConnect** or (for servers requiring SSL/TLS) **seeSmtpConnectSSL**.

4.1.1 Standard SMTP on Port 25

Standard SMTP servers do <u>not</u> use usernames or passwords. If connecting to the Internet through an ISP (Internet Service Provider), then that ISP will normally provide a SMTP server. However, if you connect to that same ISP and you are not connected to the Internet through them (say you are traveling with your laptop), then you will not normally be able to send email since this is considered "relaying".

There are several workarounds for the "relaying" problem, such as "read before send", depending on your particular ISP.

4.1.2 Standard SMTP on Port 587

Most ISPs are moving from using port 25 to using port 587. Only known servers are allowed to use port 25, which prevents connecting directly to the SMTP server associated with the email account of a recipient.

4.1.3 Authenticated SMTP on Port 25

SMTP Authentication employs a username and password to identify the user to the server. These two entities are often the same as the username and password of the corresponding POP3 (or IMAP4) account.

It is not necessary to be connected directly to your ISP in order to do SMTP Authentication. The "no relay" problem does not exist since the authentication identifies the client to the server.

4.1.4 Authenticated SMTP Port 587

This works the same as using SMTP Authentication on port 25, except that port 587 is used instead.

This is the becoming the most common used configuration.

4.1.5 TLS/SSL Authentication on Port 465 or 587

TLS and SSL are web based security protocols used by a few ISPs and most web based email services (such as Gmail, Hotmail, Yahoo mail, etc.). See section 5 "Using Stunnel". Port 465 or 587 are used. Connect with **seeSmtpConnectSSL.**

4.2 POP3 Servers

A connection to a POP3 server must be made in order to read email. However, POP3 is by far the most common protocol for reading email from an email account.

4.2.1 Standard POP3 Server on Port 110

The well known port for POP3 servers is port 110. Both an account username and password are required.

4.2.2 TLS/SSL Authentication on Port 995

TLS and SSL are web based security protocols used by a few ISPs and most web based email services (such as Gmail, Hotmail, Yahoo mail, etc.). See Section 5 "Using Stunnel". Port 995 is used. Connect with **seePop3ConnectSSL.**

4.3 IMAP Server

In order to connect to an IMAP server, the host name (or IP address), user name, and password for the IMAP account must be known.

4.3.1 Standard IMAP Server on Port 143

An IMAP server hosts your email account. It uses "well known port 143" although other ports can be used. The function **seeIntegerParam** provides the capability to set the IMAP port.

4.3.2 TLS/SSL Authentication on Port 993

TLS and SSL are web based security protocols used by a few ISPs and most web based email services (such as Gmail, Hotmail, Yahoo mail, etc.). See Section 5 "Using Stunnel". Port 993 is used. Connect with **seeImapConnectSSL.**

The **seeImapSource** function can be use to read an (undecoded) email message directly from a file. Refer to Section 6.28, "Reading Email from a File".

4.4 SMTP/POP3/IMAP Host Names

In order to send or receive email, you must know the name (or IP address) of the server. All email client programs (Eudora, Outlook, etc.) must have this name in order to send email. Note that any SMTP server that will accept your email can be used to send email.

Typically, the email server name will be `"mail.XXX.YYY"` where `XXX.YYY` is the name of the computer that hosts your email account. If you aren't sure of the email host name, look in the setup of your email client program (see Section 6.25 "Determining your SMTP Server) or ask your system administrator.

4.5 Email Address Format

Email addresses are always specified as "xxx<yyy@zzz>" where

(1) xxx is the optional "real name".

(2) yyy@zzz is the official email address, where "yyy" is the account name, and "zzz" is where the email account is hosted.

(3) The brackets are required.

For example, the email address for Technical Support can be specified by any of the following:

(1) <support@marshallsoft.com>
(2) Support< support@marshallsoft.com>
(3) Technical Support <support@marshallsoft.com>

Multiple email addresses can be strung together separated by semicolons, as in:

"<mike@myisp.com>;<pam@myisp.com>;<lauren@myisp.com>"

See the example programs for more samples of email address usage.

NOTE: All text delimiters in **SEE** were changed to the semicolon in **SEE** version 3.6.1.

5 Using Stunnel

TLS and SSL are security protocols originally developed for use with web sites, and were later adopted by web based email services such as Gmail, Hotmail, and Yahoo. They are not normally used by Internet Service Providers (ISP's).

Stunnel is a free TLS/SSL proxy server that provides TLS/SSL services to Windows programs. Stunnel is easy to install, very robust, and functions transparently with very little overhead.

Stunnel is required by SEE application programs when connecting to an email server that requires TLS/SSL services such as Gmail, Hotmail, and Yahoo.

You must use the stunnel.pem certificate file created when Stunnel version **5.35** (or above) was installed, not an older version of stunnel.pem.

Stunnel can be configured and controlled automatically (without any user intervention) if you are using SEE version 7.0 (or above). See
http://www.marshallsoft.com/stunnel.htm

Step 1: Download Stunnel

Stunnel may be downloaded from http://www.stunnel.org/downloads.html or from our FTP site at ftp://ftp.drivehq.com/MarshallSoft/PublicFolder/

Step 2: Install Stunnel

After downloading Stunnel from www.stunnel.org or from our web site, unzip the file (stunnel.zip) into your SSL directory. The recommended directory is \SEE4C\SSL (C/C++), \SEE4D\SSL (Delphi), \SEE4VB\SSL (Visual basic), etc.

Next, the Stunnel installation program must be run, which will download the required Stunnel files from www.stunnel.org as well as create a new stunnel.pem certificate file.

Step 2: Starting Stunnel

There are several ways that Stunnel can be started, although the recommended way is to put it in the StartUp list so that it is started automatically each time your computer is booted. Stunnel does not take much memory and uses CPU only when being used.

Stunnel can always be started manually by using the Windows "Run" icon or typing (for Gmail)

    c:\stunnel\stunnel config(gmail).txt

Step 3: Adding Stunnel to Startup List

First, create a shortcut to Stunnel by right-clicking on your desktop and clicking "New" then "Shortcut". For example, for Gmail, enter

    c:\stunnel\stunnel config(gmail).txt

in the box displayed. Name it "StunnelGmail".

For Windows XP;

    1. Double-click "My Computer".
    2. Double-click the Local Drive (C:) icon.
      (if you see "these files and folders are hidden", click "view these now")
    3. Scroll down to the "Documents and Settings" folder.
    4. Double-click it.
    5. Double-click "All Users"
    6. Double-click "Start Menu"
    7. Double-lick "Programs"
    8. Double-lick "Startup"
    9. Drag the Stunnel shortcut into "Startup" window.

For Windows Vista through Windows 10

    1. Right-click "Start" button
    2. Left-click "Explore All Users"
    3. Double-click "Programs"
    4. Scroll down until you see the folder labeled "Startup"
    5. Double-click it to open its contents window
    6. Drag the Stunnel shortcut into "Startup" window.

Stunnel will now be started each time your computer is booted. If more than one SSL enabled server will be used (say both Gmail and Hotmail), multiple copies of Stunnel may be started, each with its unique configuration file.

6 Application Notes

6.1 Sending Email

(1) The email client (calling **seeSmtpConnect**) connects to an SMTP server. The SMTP server can be anywhere on a TCP/IP network (or Internet), and does not require a password. However, most SMTP servers require that you be locally connected before they will accept the email for delivery.

Some "Extended SMTP" servers support "SMTP Authentication" (Section 6.10), which requires a user name and password (usually the same as your POP3 user name and password) before email can be sent. These servers also often require port 587 rather than 25.

(2) The list of email recipients is specified (**seeSendEmail** and **seeSendHTML**). The SMTP server typically verifies the domain portion of each email address (but not the full email address unless it is local), then accepts the email for delivery. After all recipients have been accepted, the email message and attachments are uploaded to the SMTP server. If no errors occur during the upload process, then the connection to the SMTP server is closed.

(3) If the email address is not on the local SMTP/POP3/IMAP server, the SMTP server connects to the recipient's SMTP server and uploads the email. The recipient's SMTP/POP3/IMAP server will verify the recipient's address and if there is a problem (such as "no such user"), the server may (at its option) send an email back to the original sender.

Refer to the **seeSendEmail** (and **seeSendHTML**) functions in the SEE Reference Manual (SEE_REF) for more details on sending email. Also refer to the MAILER example program.

6.2 Setting User Headers

One or more headers can be added by calling **seeStringParam** with parameter name SEE_ADD_HEADER. For example, the following adds two headers to all outgoing email:

```
seeStringParam(0,SEE_ADD_HEADER, "X-Priority: Normal")
seeStringParam(0,SEE_ADD_HEADER, "X-Sender: MarshallSoft Computing")
```

In order to remove all user added headers, pass a string (use SEE_SET_HEADER rather than SEE_ADD_HEADER) with the first character as a binary zero (null string).

6.3 Receiving Email

In order to receive email, you must first connect to the POP3/IMAP server. Once connected, the email messages in an account are numbered (starting with 1) and the account is locked. This number identifies email in the account. The account is not renumbered when email is deleted, but it is renumbered after the connection is closed.

Only one connection to your POP3/IMAP account is allowed at any one time. Once connected, any new arriving email is not posted until you have closed the connection.

Refer to the functions **seeGetEmailFile** and **seeGetEmailLines** in the **SEE Reference Manual** (**SEE_REF**) for more details on receiving email. Also refer to the STATUS and READER example programs.

6.4 Proxy Servers

Connecting to a proxy server is no different from connecting to any other SMTP or POP3 server. You must know the IP address of the proxy server and the user name and password assigned by the proxy server.   However, you may need to connect using a proxy specified port, rather than the standard well known SMTP and POP3 ports.   Calling seeIntegerParam with parameter SEE_SMTP_PORT or SEE_POP3_PORT before connecting can specify these alternate ports.

There is no standard method for providing connection parameters to proxy servers. Refer to the documentation for your specific proxy server.

Also see Section 5, "Using Stunnel".

6.5 Firewalls

Firewalls operate transparently with respect to TCP/IP programs, monitoring inbound and outbound traffic. Firewalls can filter packets based on their contents, source addresses, destination addresses, and port numbers. If the traffic meets criterion of the firewall, it is allowed, otherwise it is not.

6.6 CompuServe Mail

In order to check CompuServe mail, POP3 mail must first be set up. Log onto CompuServe, and execute "GO POPMAIL".

6.7 Microsoft Network (MSN)

MSN uses the authentication protocol "Secure Password Authentication", which is Microsoft proprietary. The protocol specification is not publicly available, which means that only a Microsoft email client can be used.

However, you can set an MSN account to operate with standard SMTP/POP3/IMAP servers, which will allow the use of **SEE** (as well as other standard email clients) with MSN.

6.8 Using GMAIL Email Servers

Gmail is now requiring (for some users) that email clients support the OAuth2 protocol. OAuth2 is an open standard [RFC 6749] for access delegation, commonly used as a way for Internet users to grant web sites or applications access to their data on other web sites but without revealing any passwords. TLS/SSL is still required for authentication.

SEE supports OAuth2 (beginning in version 8.2), although the user will have to acquire the "access token" that must be passed as the password to SEE. For more detail, see section 6.32 "OAUTH Delegation". or http://www.marshallsoft.com/OAUTH_Access_Delegation.htm

In addition to the above, Gmail TLS/SSL security settings may need to be edited. See https://www.google.com/settings/security/lesssecureapps

Gmail doesn't recognize periods as characters within usernames. Remove all periods from user names passed to SEE functions. See https://support.google.com/mail/answer/10313?hl=en

6.9 Using HOTMAIL/LIVE and YAHOO Email Servers

If you have an account at Hotmail/LIVE, Yahoo, or any other server that requires the use of the SSL protocol, it is possible to send and receive email using an SSL enabled proxy server such as the (free) Stunnel proxy server. Refer to Section 5.0 "Using Stunnel".

6.10 Secure Email

There are several ways to implement secure email using **SEE**. The first way is to use a package such as "Pretty Good Privacy" (search http://www.download.com for "PGP") to encrypt the message, optionally compress the resulting file using PKZIP or similar product, and then send the encrypted message as an attachment.

Another way to send encrypted messages with PGP is to first encrypt the message using PGP (which yields 7-bit ASCII text) then paste it directly into the message portion of the outgoing email. Once downloaded by the recipient, the encrypted portion of the message is decrypted using PGP.

A "Virtual Private Network" (VPN) can also be created which automatically encrypts all data (not just email) sent over the VPN. VPN products may be either software or hardware, and are widely available. If running Windows XP, enter "VPN" in the search box of Windows HELP and SUPPORT.

6.11 SMTP Authentication

SMTP authentication requires connection to an ESMTP (Extended SMTP) server. Three forms of SMTP authentication are supported: "AUTH PLAIN", "AUTH LOGIN" and "AUTH CRAM-MD5".

"AUTH PLAIN" is fairly simple protocol supported by many SMTP servers that support authentication.

"AUTH LOGIN" is a Microsoft authentication mechanism used in Exchange Server. It is not an Internet standard nor is it covered by an RFC.

"AUTH CRAM-MD5" is based on public key encryption and is the preferred authentication protocol. It is covered by RFC2554 and RFC 2195.

To perform SMTP authentication, make the following calls before connecting:

```
seeIntegerParam(0, SEE_ENABLE_ESMTP, 1)
seeStringParam(0, SEE_SET_USER,    your-user-name)
seeStringParam(0, SEE_SET_SECRET,  your-password)
```

The 'secret' is normally the same as the user's password.

A particular protocol can be specified calling

```
seeIntegerParam(0, SEE_AUTHENTICATE_PROTOCOL, X)
```

where "X" is set to AUTHENTICATE_PLAIN, AUTHENTICATE_LOGIN, AUTHENTICATE_CRAM, or AUTHENTICAT_XOAUTH2

6.12 POP3 Authentication

POP3 authentication is performed by the use of either (1) the USER and PASS commands, or the (2) APOP command. Since the APOP command is an optional command, some POP3 servers have not implemented it.

To perform APOP authentication rather than USER/PASS, make the following call before connecting:

```
seeIntegerParam(0, SEE_ENABLE_APOP, 1)
```

6.13 MIME Extensions

Internet mail can only transport 7-bit ASCII characters. Multipurpose Internet Mail Extensions (MIME) is used to allow the attachment of binary data to an email message.

The standard MIME attachment types are "quoted-printable" and "base64". The **SMTP/POP3/IMAP Email Engine** library supports both.

6.13.1 Quoted-Printable Encoding

Quoted-Printable encoding is used for three primary purposes:

(1) To embed binary values into an email message, typically for use with foreign alphabets (characters).
(2) To send email messages wider than 78 characters.
(3) To embed HTML text into the email message.

To enable Quoted-Printable encoding, call

```
seeIntegerParam(Chan, SEE_QUOTED_PRINTABLE, QUOTED_PLAIN)
```

before calling **seeSendEmail**. To embed HTML text into an email message, which can be rendered by email clients capable of interpreting HTML (such as Outlook Express), call

```
seeIntegerParam(Chan, SEE_QUOTED_PRINTABLE, QUOTED_HTML)
```

To allow ISO-8859-1 characters in an email message, call

```
seeIntegerParam(Chan, SEE_QUOTED_PRINTABLE, QUOTED_8859)
```

To disable Quoted-Printable encoding (the default), call

```
seeIntegerParam(Chan, SEE_QUOTED_PRINTABLE, QUOTED_OFF)
```

6.13.2 Binary Attachments

Binary attachments are encoded using MIME base-64. Most email clients (such as made by Eudora, Netscape, and Microsoft) can decode MIME base-64 attachments.

To attach a file to an email, specify the filename as the last argument of the **seeSendEmailFile** function. Refer to the **SMTP/POP3/IMAP Email Reference Manual** (SEE_REF) for more details. Multiple attachments are listed with semicolons separating them, such as "file1.zip;file2.zip;file3.zip".

Only named attachments are decoded. To specify that unnamed attachments also be decoded, call

```
seeIntegerParam(0, SEE_DECODE_UNNAMED, 1)
```

before calling **seeGetEmailFile**.

6.14 ISO-8859 Character Sets

ISO 8859 is a full series of 10 standardized multilingual single-byte coded (8bit) graphic character sets for writing in alphabetic languages:

```
ISO-8859-1  Latin1 (West European)       ISO-8859-6  Arabic
ISO-8859-2  Latin2 (East European)       ISO-8859-7  Greek
ISO-8859-3  Latin3 (South European)      ISO-8859-8  Hebrew
ISO-8859-4  Latin4 (North European)      ISO-8859-9  Turkish
ISO-8859-5  Cyrillic                     ISO-8859-10 Nordic
```

The graphic character assigned to each of the upper 128 bytes (0x80 thru 0xff) are uniquely specified for each individual ISO character set. The lower 128 bytes (0x00 thru 0x7f) are the standard 7-bit ASCII assignments.

In order to specify ISO-8859-1for outgoing email, call (before sending email)

```
seeIntegerParam(0, SEE_QUOTED_PRINTABLE, QUOTED_ISO_8859_1);
```

For example;

```
// C/C++ Example:
seeIntegerParam(0,SEE_QUOTED_PRINTABLE, QUOTED_ISO_8859_1);
```

SEE directly supports ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, ISO_8859_7, and ISO_8859_8.

Also refer to http://www.marshallsoft.com/iso-8859.htm

6.15 Windows 1252 and 1255 Character Sets

The Windows 1250, 1252, and 1255 character sets can be specified as follows:

```
seeIntegerParam(0, SEE_QUOTED_PRINTABLE, QUOTED_WIN_1250); // set Windows-1250
seeIntegerParam(0, SEE_QUOTED_PRINTABLE, QUOTED_WIN_1252); // set Windows-1252
seeIntegerParam(0, SEE_QUOTED_PRINTABLE, QUOTED_WIN_1255); // set Windows-1255
```

6.16 16-bit Character Sets

Any language text that is represented by 16-bit character codes (such as Chinese, Japanese, and Korean) can be inserted into email. For example, if the body of the email contains GB2312 characters (16-bit binary Chinese character codes), the character set can be specified in the email as (C/C++ example)

```
Code = seeIntegerParam(0, SEE_QUOTED_PRINTABLE, QUOTED_USER);
Code = seeStringParam(0, SEE_SET_CONTENT_TYPE,
       (char *)"Content-Type: text/plain; charset=gb2312");
```

The above approach works for any defined character set such as

```
gb2312         Simplified Chinese
big5           Chinese
euckr          Korean
koi8-r         Cyrillic
ms_kanji       Japanese
```

6.17 Unicode & UTF-8

Unicode is a computing industry standard "universal" character set in which each character graphic is assigned one 16-bit value. Character graphics from most of the world's writing systems are represented in the Unicode standard.

See http://www.unicode.org for more information on the Unicode standard.

UTF-8 is a variable length 8-bit byte encoding of Unicode. Any 16-bit Unicode text can be encoded into UTF-8 text by calling the **seeEncodeUTF8** function. Conversely, UTF-8 encoded text can be decoded into 16-bit Unicode by calling the **seeDecodeUTF8** function.

Also see the CODETEST example program.

6.18 Wide Text

Wide text refers to character s that are encoded in 16-bits rather than 8 bits and is used to encode 16-bit Unicode.

Note however, that the recipient's email client must be capable of decoding UTF-8 and displaying 16-bit Unicode characters.

According to RFC 822, all email messages must be formatted as 7-bit ASCII text with each line ending in a carriage return line feed pair. Each line of text should be no more than 1000 bytes.

In order to allow wider email messages, call

```
seeIntegerParam(Chan, SEE_QUOTED_PRINTABLE, QUOTED_PLAIN)
```

before sending email. Also refer to Section 6.12.1 "Quoted-Printable Encoding".

6.19 Embedded HTML

Graphic images may be embedded within HTML encoded email by passing the image filename(s) in the **seeSendHTML** function. Within the HTML encoded email itself, the first image must be referenced as:

```
<IMG SRC="cid:message-root.1">
```

Additional images may be reference as "message-root.2", "message-root.3", etc. Note that the "cid:message-root" above must be in lower case.

See the HTML (or SendHTML.C for SEE4C) example program.

6.20 Attaching Graphic Files

Attaching files ending with ".BMP", ".GIF", ".TIF", and ".JPG" are attached as image types, and therefore can be displayed by email clients (such as Eudora and Outlook, etc.) that are capable of displaying graphics files. To disable this feature, call

```
seeIntegerParam(Chan, SEE_ENABLE_IMAGE, 0)
```

To enable this feature, call

```
seeIntegerParam(Chan, SEE_ENABLE_IMAGE, 1)
```

6.21 Downloading Attachments

Specify a download directory when calling **seeGetEmailFile** so that you don't overwrite an existing file of the same name in the current directory. This is an important security precaution.

For example (double backslashes required for C/C++ only):

```
seeGetEmailFile(Chan,MsgNbr,MsgName,".\\download",".\\download")
```

After downloading, the list of attachment filenames can be found by calling

```
seeDebug(Chan,SEE_GET_ATTACH_NAMES,Buffer,BufferLength)
```

6.22 Message Status

The POP3 server typically inserts the header line

```
Status: U
```

in the header area of newly received messages. Once the email message has been read, the POP3 server changes this to

```
Status: R
```

6.23 Return Receipt Requested

The SMTP protocol itself has no provisions for return receipt acknowledgements. However, most SMTP servers recognize several headers. Older SMTP servers may recognize the header

```
Return-Receipt-To:
```

while newer SMTP servers will recognize the header:

```
Disposition-Notification-To:
```

For example:

```
seeStringParam(Chan, SEE_ADD_HEADER,
   "Disposition-Notification-To: <info@marshallsoft.com>")
```

You can also send both headers.

6.24 Verifying Users

The **seeVerifyUser** function can be used to verify an email account, as demonstrated in the VERUSR example program.

However, some SMTP servers may refuse to connect to non-local clients. Those that do may refuse to honor the verify request. This means that a negative verify response does NOT mean that the email address is necessarily incorrect.

6.25 Connection Status

If **seeStatistics**(Chan, SEE_GET_CONNECT_STATUS) is called and it fails because there is no live TCP/IP network, **seeRelease** must be called (followed by calling **seeAttach** again) to re-initialize the winsock layer. The problem appears to be in the Microsoft winsock code.

Rather than use SEE_GET_CONNECT_STATUS, attempt to connect to any (SMTP or POP3) email server and send the NOOP command (with **seeCommand**):

```
Code = seeCommand(Chan, "NOOP")
```

6.26 Determining Your SMTP Server

All email clients (such as Outlook, Eudora, .etc) require a SMTP server host name (or IP address) before they can send email. If an email client is installed on a computer, the SMTP that it uses can be found. For example:

Eudora: Tools/Options/Sending Mail/SMTP Server:

Outlook Express: Tools/Accounts/Mail/Properties/Servers/Outgoing mail (SMTP)

Outlook: Tools/Options/Mail Setup/E-mail Account/View.../[Next]/Change.../Outgoing mail server (SMTP)

Mozilla Thunderbird: Tools/Account Settings.../Outgoing Server (SMTP)/Server Name:

Netscape Mail: Window/Mail & Newsgroups/Edit/Mail & Newsgroups Account Settings.../Outgoing Server (SMTP)/Server Name:

6.27 Undecoded Email

There are several ways to get an undecoded copy of (incoming) email. **seeGetEmailLines** can be called or an undecoded copy can be saved to disk when **seeGetEmailFile** is called.

In the latter case, if the function

```
    seeStringParam(Chan,  SEE_SET_RAWFILE_PREFIX,  prefix-character)
```

is called before calling **seeGetEmailFile**, then an undecoded copy of the email being downloaded will be saved to disk whose name consists of the email file name (3rd argument in **seeGetEmailFile**) prefixed by the above specified prefix character.

For example, if the underscore character "_" is specified as the prefix character, and "Email.txt" is specified in **seeGetEmailFile** for the message file, then the undecoded copy of the downloaded email will be save to "_Email.txt" in the same directory to which the message file is written.

See the entry for **seeGetEmailLines**, **seeGetEmailFile**, and **seeStringParam** (with parameter SEE_SET_RAWFILE_PREFIX) in the SEE Reference Manual (SEE_REF.*) and also the READER example program.

6.28 Forwarding Email

In order to forward email, you must have an undecoded copy of the email to be forwarded. To get an undecoded copy, refer to Section 6.26 "Undecoded Email".

The **seeForward** function forwards email by using the "message/rfc822" content type.

Call **seeForward** to forward an email to a new recipient. See the entry for **seeForward** in the SEE Reference Manual (SEE_REF.*) and the FORWARD example program.

6.29 Reading Email from a File

The **seePop3Source** function can be used to specify the filename of an undecoded email message so that it can be read (and thus decoded) when **seeGetEmailFile** is called.

Rather than connecting to a POP3 server and reading a specific email message from the server, the function **seePop3Source** is called. Subsequently calling **seeGetEmailFile** will read and decode the email as if it were being downloaded from a POP3 server.

See the entry for **seePop3Source** in the SEE Reference Manual (SEE_REF) and the POP3RD example program.

6.30 Sending SMS Messages

Most (if not all) cell phone companies provide an email gateway server that forwards email to a cell phone. In order to send an SMS message with an email client such as SEE, you must know both the cell phone number and the cell phone company.

Up to 160 characters can be sent, consisting of the subject plus the body of the email (cut off at 160 characters).

Some of the more popular cell phone companies SMS email addresses are:

```
ALLTEL         - 10DigitPhoneNumber@message.alltel.com
AT&T Wireless  - 10DigitPhoneNumber@mobile.att.net
Cingular       - 10DigitPhoneNumber@mobile.mycingular.com
Fido           - 10DigitPhoneNumber@fido.ca
Virgin Mobile  - 10DigitPhoneNumber@vmobl.com
Boost Mobile   - 10DigitPhoneNumber@myboostmobile.com
Sprint PCS     - 10DigitPhoneNumber@messaging.sprintpcs.com
T-Mobile       - 10DigitPhoneNumber@tmomail.net
Verizon        - 10DigitPhoneNumber@vtext.com
Cricket        - 10DigitPhoneNumber@mms.mycricket.com

Example: 8885551234@mobile.att.net
```

Most cell phone company email-to-SMS gateway servers can be found online without too much trouble. For example, to find the gateway server for Verizon, search google for "verizon sms gateway".

6.31 Character Quoting

The content of all email messages, including attachments, must be 7-bit ASCII text. In addition, certain characters (such as the equals sign '=') are not allowed in HTML formatted email messages.

"Quoting" is a method used to escape characters in an email message that would otherwise not be legal. For character codes 00 thru 7F (hex), a non-legal character would be replaced by its hexadecimal character code equivalent. For example, the equals sign (not legal in HTML email messages) would be replace by the 3 characters "=3D".

For character codes above 7F, the escape sequence depends on the selected character set. For example, the word "français" in ISO-8859-1 would be replaced by "fran=E7ais" since E7 is the hexadecimal character code for 'ç' in ISO-8859-1.

Character sets supported by SEE include:

```
QUOTED_HTML         QUOTED_ISO_8859_1     QUOTED_ISO_8859_2
QUOTED_ISO_8859_3   QUOTED_ISO_8859_4     QUOTED_ISO_8859_7
QUOTED_ISO_8859_8   QUOTED_PLAIN
QUOTED_RICH         QUOTED_UTF8
QUOTED_WIN_1250     QUOTED_WIN_1252       QUOTED_WIN_1255
```

Quoting is enabled by calling `seeIntegerParam(Chan, SEE_PRINTABLE_QUOTED, N)` where N is one of the above character sets (`QUOTED_HTML`, etc).

6.32 OAUTH2 Delegation

OAuth2 is an open standard [RFC 6749] for access delegation, commonly used as a way for Internet users to grant web sites or applications access to their information on other web sites but without giving them the passwords. That is, OAuth2 manages access authorization (delegation) not access authentication.

OAuth2 is designed to allow someone else access to one's SMTP, POP3, or IMAP server. Security is still handled by SSL/TLS.

See https://en.wikipedia.org/wiki/OAuth (the "Controversy" section is quite interesting)
and https://tools.ietf.org/html/rfc6749 (OAuth 2.0 Authorization Framework)

Note that, in all cases, the delegation token **must** be provided by the user. SEE does **not** connect to token servers.

How OAuth2 works (GMail Example)

1. You must register your application program in the Google API Console.
2. When your application launches, it requests that the user grant access to data in their Google account.
3. If the user consents, your application requests and receives an "access token" to access the Gmail API.

For more details, see:

 https://developers.google.com/gmail/api/
 https://developers.google.com/gmail/imap/imap-smtp
 https://developers.google.com/gmail/imap/xoauth2-protocol
 https://google.github.io/google-api-cpp-client/latest/guide/oauth2.html
 https://docs.microsoft.com/en-us/exchange/client-developer/legacy-protocols/how-to-authenticate-an-imap-pop-smtp-application-by-using-oauth

For SMTP applications, the OAUTH2 protocol must first be specified by calling:

```
  seeIntegerParam(0, SEE_AUTHENTICATE_PROTOCOL, AUTHENTICATE_XOAUTH2)
```

For all SMTP, POP3, and IMAP4 applications that use OAUTH2, the delegation token (that the user must supply) must be encoded into a password string by calling **seeMakeXOAuth2()**, after which the created password string is passed as the password argument in **seeSmtpConnectSSL(), seePop3ConnectSSL()** or **seeImap4ConnectSSL()**.

Also see section 6.34 "Two Factor Authentication", which is an alternative to OAUTH2 delegation.

6.33 Free SMTP Servers

There are quite a few SMTP servers on the net that offer a free plan. Google "public smtp servers" to see what is available.

We suggest one that supports "SMTP Authentication" but does not require TLS/SSL. For example:

```
https://sendgrid.com/

https://www.smtp2go.com/
```

Note that TLS/SSL (used by most national web based email services such as Gmail and Yahoo) only provides encryption between your client program and their SMTP server, not all the way through to the recipient. Email should never be considered secure, regardless of the email client or the SMTP server being used.

6.34 Two Factor Authentication

The major Web-Mail providers (GMail, Yahoo, Outlook, Office 365, etc.) support "two factor authentication" (2FA) and "app passwords" for use by third party email clients (such as SEE programs). The app password is a large randomly generated password string that is used exactly like the account password when connecting to Web-Mail SMTP, POP3, and IMAP servers.

The procedure is:

(1) Log onto your (Gmail, Yahoo, Office 365, etc) account.
(2) Set up two factor authentication.
(3) Generate an "app password", which may then be used in SEE applications.

```
The password for the app is a 16 character code generated on your device. App
passwords can only be used with accounts that have enabled two factor
authentication.
```

6.34.1 GMail

```
https://knowledge.workspace.google.com/kb/how-to-create-app-passwords-000009237

1. Log into your Google account: https://myaccount.google.com/.
2. Select Security.
3. In the "How to access Google" section, select Two-step verification.
4. At the bottom of the page, select Password for apps.
5. Enter a name.
6. Select Generate.
7. Select End.
```

6.34.2 Yahoo

```
https://help.yahoo.com/kb/SLN15241.html

1. Log into your Yahoo Accoun: https://login.yahoo.com/
2. Select Security.
3. Select Generate app password under the section Other ways to sign in.
4. On the App passwords pop-up, enter the name of the App (Drake Tax) and click
Generate password.
5. A window will display your App password. Click Copy and continue below.
6. Do not click Done until you have copied the password.
```

6.34.3 Office 365

```
https://support.microsoft.com/en-us/account-billing/manage-app-passwords-for-
two-step-verification-d6dc8c6d-4bf7-4851-ad95-6d07799387e9

1. Sign in to your work or school account
2. Go to the My Account page
3. Select Security info from the left navigation pane
4. Select Add method
5. Select App password from the list
6. Select Add
7. Enter a name for the app password
8. Select Next
9. Copy the password from the App password page
10. Select Done
```

7 Theory Of Operation

The **SMTP/POP3/IMAP Email Engine** component library is state driven. This means that each call to **SEE** functions (that access the server) is broken down into sequential steps, each of which can be performed within a second or two. There are two ways in which **SEE** is used: (1) indirect use of the state engine, and (2) direct use of the state engine.

7.1 Indirect Method

The first (or "indirect") way to use the **SEE** library is to allow all **SEE** function calls to automatically call the SEE driver (**seeDriver**) before returning. This is the default way that **SEE** library operates.

The major advantage of this approach is that each **SEE** function returns only after it has completely finished. The disadvantage of this approach is that some functions may run for a considerable amount of time during which time the calling application must wait.

Refer to the sample programs MAILER and STATUS for an example of this approach.

7.2 Direct Method

The second (or "direct") way that the **SEE** state driver is used is to call it (**seeDriver**) directly. In order to operate this way, the function **seeIntegerParam** must be called which sets the AUTO_CALL flag to off:

```
seeIntegerParam(Chan, SEE_AUTO_CALL_DRIVER, 0)
```

After the above statement is executed, the state driver (**seeDriver**) must be called after all other **SEE** functions that access the server. For example (pseudocode example),

```
... enable direct mode (disable indirect mode).
seeIntegerParam(Chan, SEE_AUTO_CALL_DRIVER, 0)
... connect to server.
Code = seeSmtpConnect(...)
If Code < 0 Then
 ... handle error here.
End If
... run the driver.
Loop
  ... call the driver
  Code = seeDriver(Chan)
  If Code < 0 Then
    ... handle error here.
    Exit Loop
  End If
  If Code = 0 Then
    ... seeDriver has finished.
    Exit Loop
  End If
  ... display progress or do other processing here.
End Loop
... enable indirect mode (disable direct mode).
seeIntegerParam(Chan, SEE_AUTO_CALL_DRIVER, 1)
```

The major advantage of the direct approach is that the calling application can perform other work such as reporting the progress of large downloads. The disadvantage is the extra code that must be written to call **seeDriver**.

Refer to the sample program READER (or STATUS) for an example of this approach.

8 Using SEE with Other Languages

We have versions of the **SEE** toolkit for C/C++ (.NET and C# .NET) (SEE4C), Delphi (SEE4D), Visual Basic (VB .NET. VBA) (SEE4VB), PowerBASIC (SEE4PB), Visual FoxPro (SEE4FP), Visual dBase (SEE4DB), Alaska Xbase++ (SEE4XB), COBOL (SEE4CB), and Fortran (SEE4F). All versions of the **SEE** library use the same DLLs (SEE32.DLL or SEE64.DLL). Evaluation versions for these may be downloaded from our website at

> http://www.marshallsoft.com/email-component-library.htm

The **SMTP/POP3/IMAP Email Engine** DLL can also be used with any Win32 application written in any language capable of calling the Windows (95/98/Me/NT/2000/2003/2012/XP/Vista/Win7/Win8) API.

8.1 Using SEE with Supported Languages.

Once you have purchased one programming language version of the **SMTP/POP3/IMAP Email Library SDK (SEE)**, you can use it with all other supported languages. Supported languages are C/C++, .NET, Visual Basic, PowerBASIC, Delphi, Visual FoxPro, Visual dBase, Xbase++, and (Fujitsu) COBOL.

For example, assume that you have previously downloaded and installed the registered version of SEE4C and now you want to also call **SEE** functions from Visual Basic.

1. Make a backup copy of SEE32.DLL found in the Windows directory (\WINDOWS or \WINNT).
2. Download and install the evaluation version of SEE4VB (http://www.marshallsoft.com/see4vb.htm)
3. Compile and run the Visual Basic SEEVER example program found in the APPS directory created in step 2 above. It should display the pop-up screen.
4. Restore SEE32.DLL saved in step 1 above.
5. Paste the key code value found in (the registered version of) KEYCODE.H into KEYCODE.BAS.
6. Run the Visual Basic SEEVER example program again. It should no longer display the pop-up screen.

A quicker and easier way would be to request multiple programming versions of SEE when a developer license is purchased. There is no additional charge.

8.2 Using SEE with Unsupported Languages

The **SMTP/POP3/IMAP Email Engine** DLLs can be used with any application written in any language capable of calling the Windows (95/98/ME/XP/2003/2012/Vista/Win7/Win8/NT/2000) API. SEE32.DLL is required for all Win32 applications and SEE64.DLL is required for Win64 applications.

The following programming languages have defined declaration files:

```
C/C++                   SEE.H
Visual Basic            SEE32.BAS
VBA (Excel, Access,…)   SEE32.BAS
PowerBASIC              SEE32.BAS [not the same as above]
Codegear Delphi         SEE32.PAS
Fujitsu COBOL           SEE32.CBI
ABSOFT FORTRAN          SEE32ABS.INC
Compaq Visual Fortran   SEE32DEC.INC [formerly Digital Visual Fortran]
Visual FoxPro           SEE32.FOX
Visual dBase            SEE32.CC
Alaska Xbase++          SEE32.CH
```

Additional declaration files will be added. Give us a call if you need a declaration not listed above.

If you have interfaced **SEE** to an unusual programming language, email us the declaration file!

9 Versions of the SMTP/POP3/IMAP Email Engine

The **SMTP/POP3/IMAP Email Engine** (SEE) SDK library is available in three versions. All three versions have identical functionality.

9.1 Evaluation Version

The evaluation version can be differentiated from the other two versions by:

(1) The registration reminder screen is displayed at startup and every 5 minutes thereafter.

(2) The "X-OEM: " header in all outgoing email is branded with

```
"X-OEM: EVALUATION VERSION [http://www.marshallsoft.com]"
```

(3) Each email is followed by the following two lines:

```
"_____"
"MarshallSoft SMTP/POP3 Engine. Programmers see www.marshallsoft.com"
```

The evaluation version may <u>not</u> be used for commercial purposes.

(4) The evaluation version stops working after 30 days.

9.2 Academic Version

The academic version can be differentiated from the other two versions by:

(1) There is no registration reminder screen.

(2) The "X-OEM: " header in all outgoing email is branded with

```
"X-OEM-To: ACADEMIC INSTITUTE [http://www.marshallsoft.com]"
```

(3) There are no lines added to the end of the email as in the evaluation version. That is, the lines shown above in section 8.1 (3) are <u>not</u> present in the academic version.

DLL's purchased with the academic discount may not be distributed, and must be used for educational purposes only.

9.3 Professional Version

The professional version can be differentiated from the other two versions by:

(1) There is no registration reminder screen.

(2) The "X-OEM: " header in all outgoing email is branded with your company name or Customer ID, although most email clients do not display this.

(3) There are no lines added to the end of the email as in the evaluation version. That is, the lines shown above in Section 8.1 (3) are <u>not</u> present in the professional version.

Your compiled DLL may be distributed with your compiled applications as specified by the software license. However, the Keycode to the DLL cannot be distributed. The Professional version may be used for commercial purposes. Licensing information is provided in Section 11.1

10 Resolving Problems

(1) First, be sure you are passing the proper key code. See Section 2.2, "Keycode". Before attempting to run any of the example programs, you should already be able to connect to the Internet (or TCP/IP LAN) and run your email client program, such as Outlook, Eudora, or Pegasus Mail.

(2) If the registration reminder screen (popup) is still being displayed after purchasing a license, the problem is that Windows is finding the evaluation version of the DLL before the registered DLL. The solution is to delete (or zip up) all evaluation versions of SEE32.DLL or SEE64.DLL. Run SETUP and then recompile.

(3) If "error -74" is received when calling **seeAttach**, the problem is that the keycode passed to **seeAttach** does not match the keycode in the DLL. This is caused by (1) using the evaluation keycode (value = 0) with the registered DLL, or (2) using the registered keycode with the evaluation DLL.

(4) If you have trouble connecting to a SMTP or POP3 server, try using the IP address instead of the server name. If using the IP address works but the server name does not, the problem lies with the Domain Name System (DNS) lookup. If this does not solve the connection problem, try connecting using TELNET, located in the Windows directory. Use port 25 for SMTP, 110 for POP3, and 143 for IMAP. Also, make sure that you have edited the file EMAIL.* with your ISP parameters.

(5) If you cannot get your application to run properly, first compile and run the example programs. If you call us to report a possible bug in the library, the first thing we will ask is if the example programs run correctly.

(6) Be sure to test the code returned from **SEE** functions. Then call **seeErrorText** to get the text associated with the error code. (The file, ERRORS.TXT, contains a list of all error codes.)

For example (C Example):

```
Code = seeSmtpConnect(0,"mail.isp.net","<support@marshallsoft.com>",NULL);
if(Code<0)
  {static char Buffer[64];
   seeErrorText(0,Code,Buffer,64);
   printf("Error %d: %s\n", Code, Buffer);
  }
```

Another good idea is turn on logging by calling

seeStringParam(Chan, SEE_LOG_FILE, logfilename)

We recommend the following steps if you believe that you have discovered a bug in the library:

(1) Create the smallest, simplest test program possible that demonstrates the problem.
(2) Document your exact machine configuration and what error the test program demonstrates.
(3) Email to **support at marshallsoft.com** the example source and log file.

If the problem is an error in the library and can be solved with an easy work-around, we will publish the work-around. If the problem requires a modification to the library, we will make the change and make the modified library available to our customers without charge.

11 Legal Issues

11.1 License

This license agreement (LICENSE) is a legal agreement between you (either an individual or a single entity) and MarshallSoft Computing, Inc. for this software product (SOFTWARE). This agreement also governs any later releases or updates of the SOFTWARE. By installing and using the SOFTWARE, you agree to be bound by the terms of this LICENSE. If you do not agree to the terms of this LICENSE, do not install or use the SOFTWARE

MarshallSoft Computing, Inc. grants a nonexclusive license to use the SOFTWARE to the original purchaser for the purposes of designing, testing or developing software applications. Copies may be made for back-up or archival purposes only. This product is licensed for use by only one developer at a time. All developers working on a project that includes a MarshallSoft Software SDK, even though not working directly with the MarshallSoft SDK, are required to purchase a license for that MarshallSoft product.

The "academic" registered DLL's may not be distributed under any circumstances, nor may they be used for any commercial purpose.

The "professional" registered DLL's may be distributed (royalty free) in object form only, as part of the user's compiled application provided the value of the Keycode is not revealed. The registered DLL's may NOT be distributed as part of any software development system (compiler or interpreter) without our express written permission.

Note that registered DLL's do not expire. Registered users may download free updates for a period of one year from the date of purchase.

11.2 Warranty

MARSHALLSOFT COMPUTING, INC. DISCLAIMS ALL WARRANTIES RELATING TO THIS SOFTWARE, WHETHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND ALL SUCH WARRANTIES ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. NEITHER MARSHALLSOFT COMPUTING, INC. NOR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION, OR DELIVERY OF THIS SOFTWARE SHALL BE LIABLE FOR ANY INDIRECT, CONSEQUENTIAL, OR INCIDENTAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE SUCH SOFTWARE EVEN IF MARSHALLSOFT COMPUTING, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR CLAIMS. IN NO EVENT SHALL MARSHALLSOFT COMPUTING, INC.'S LIABILITY FOR ANY SUCH DAMAGES EVER EXCEED THE PRICE PAID FOR THE LICENSE TO USE THE SOFTWARE, REGARDLESS OF THE FORM OF THE CLAIM. THE PERSON USING THE SOFTWARE BEARS ALL RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE.

Some states do not allow the exclusion of the limit of liability for consequential or incidental damages, so the above limitation may not apply to you.

This agreement shall be governed by the laws of the State of Alabama and shall inure to the benefit of MarshallSoft Computing, Inc. and any successors, administrators, heirs and assigns. Any action or proceeding brought by either party against the other arising out of or related to this agreement shall be brought only in a STATE or FEDERAL COURT of competent jurisdiction located in Madison County, Alabama. The parties hereby consent to in personam jurisdiction of said courts.

12 SEE Function Summary

Refer to the **SMTP/POP3/IMAP Email Reference Manual** (SEE_REF) for detailed information on the **SEE** functions. There are 68 functions in the **SMTP/POP3/IMAP Email Engine** component library.

```
seeAbort            Aborts SEE session.
seeAttach           Attaches SEE DLL.
seeAttachmentParams Specify attachment content parameters.
seeByteToShort      Converts 8-bit character buffer to 16-bit.
seeClose            Closes SMTP/POP3/IMAP Email Engine.
seeCommand          Transmit arbitrary SMTP/POP3/IMAP command.
seeConfigSSL        Adds lines to SSL configuration file.
seeDebug            Returns debug information.
seeDecodeBuffer     Decodes base-64 buffer.
seeDecodeUTF8       Decodes UTF-8 character to a 16-bit Unicode character.
seeDecodeUTF8String Decodes UTF-8 string to a 16-bit Unicode string.
eeDecodeUU          Decodes UU-encoded text.
seeDeleteEmail      Deletes email.
seeDriver           Executes next SEE state.
seeEncodeBuffer     Encodes base-64 buffer.
seeEncodeUTF8       Encodes 16-bit Unicode character to a UTF-8 character.
seeEncodeUTF8String Encodes 16-bit Unicode string to a UTF-8 string.
seeErrorText        Get text associated with error code.
seeExtractLine      Extracts line by line number.
seeExtractText      Extracts line containing specified text.
seeForward          Forward an email.
seeForward          Forwards email.
seeGetEmailCount    Get number of emails waiting on server.
seeGetEmailFile     Read email file and save to disk.
seeGetEmailLines    Read email lines into buffer.
seeGetEmailSize     Get size of email message on server.
seeGetEmailUID      Get email message user ID string.
seeGetHeader        Get email header.
seeImapConnect      Connects to IMAP server.
seeImapCopyMBmail   Copies mail between IMAP mailboxes.
seeImapCreateMB     Creates a mailbox.
seeImapDeleteMB     Deletes a mailbox.
seeImapFlags        Get, set, or delete IMAP message flags.
seeImapListMB       List all IMAP mailboxes.
seeImapMsgNumber    Gets message #'s from buffer filled by seeImapSearch
seeImapRenameMB     Renames an IMAP mailbox.
seeImapSearch       Search for IMAP messages with specified flags.
seeImapSelectMB     Selects IMAP mailbox.
seeImapSource       Specifies file from which to read IMAP email.
seeIntegerParam     Parameters to control how email is sent/receive.
seeIsConnected      Tests if still connected to the email server.
seeKillProgram      Kill (terminate) external program.
seeMakeSubject      Makes quoted subject string.
seeMakeXOAuth2      Makes XOAUTH2 string.
seePop3Connect      Connects to POP3 server.
seePop3ConnectSSL   Connect to POP3 server via SSL proxy server.
seePop3Source       Specifies file from which to read POP3 email.
seeQuoteBuffer      Creates ISO-8859 encoded strings.
seeReadQuoted       Reads (and quotes) a file to a buffer.
seeRelease          Releases SEE.
seeSendEmail        Sends email and attachments.
seeSendHTML         Sends HTML encoded email.
seeSetFileBuffer    Pass buffer (not file) as attachment.
seeShortToByte      Converts 16-bit ASCII character buffer to 8-bit.
seeSetCertAuth      Specifies Certificate Authority (CA) directory and file.
seeSetErrorText     Sets error message language (French, German, etc.)
seeSleep            Sleep specified milliseconds.
```

```
seeSmtpConnect       Connects to SMTP server.
seeSmtpConnectSSL    Connect to SMTP server via SSL proxy server.
seeSmtpTarget        Specifies SMTP output file.
seeStartProgram      Start external program.
seeStatistics        Returns runtime statistics.
seeTestConnect       Tests connection to a server.
seeStringParam       Sets SEE string parameter for email control.
seeUnQuoteBuffer     Un-quotes buffer.
seeVerifyFormat      Check email address format.
seeVerifyUser        Check email address with SMTP server.
```