SMTP/POP3/IMAP Email Engine

Library for Delphi

Programmer's Manual

(SEE4D)

Version 8.4

February 23, 2023

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

MarshallSoft Computing, Inc.
Post Office Box 4543
Huntsville AL 35815

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

TABLE OF CONTENTS

1 Introduction

The **SMTP/POP3/IMAP Email Engine for Delphi** (**SEE4D**) library is toolkit that allows software developers to quickly develop SMTP and POP3/IMAP email applications in Delphi and Delphi for .NET.

A straightforward interface allows sending and receiving email, including multiple MIME base64 and quoted-printable encoded attachments, over any TCP/IP network (such as the Internet). Knowledge of Winsock and TCP/IP is not needed.

The **SMTP/POP3/IMAP Email Engine (SEE)** SDK is a component library that uses the Windows API to provide direct and simple control of the SMTP (Simple Mail Transport Protocol), POP3 (Post Office 3), and IMAP 4 (Internet Message Access Protocol) protocols.

The **SMTP/POP3/IMAP Programmer's Manual for Delphi** provides information needed to compile and run programs in a Delphi programming environment.

The **SMTP/POP3/IMAP Email Engine for Delphi (SEE4D)** component library supports and has been tested with all 32-bit and 64-bit versions of Delphi including:

- Borland Delphi (2.0, 3.0, 4.0, 5.0. 6.0 and 7.0)
- Borland Delphi 8 for .NET
- Borland Delphi 2005 & 2006
- Borland Turbo Delphi
- Codegear Delphi 2007
- Embarcadero Delphi 2009 & 2010
- Embarcadero Delphi XE through XE10 & Seattle

SEE4D includes numerous example programs that demonstrate SMTP and POP3/IMAP functions used to create mail applications using the **SEE** library.

SEE4D runs under all versions of 32-bit and 64-bit Windows through Windows 10. The **SMTP/POP3/IMAP Email Engine SDK** DLLs (SEE32.DLL or SEE64.DLL) can also be used from any language (C/C++, .NET, Visual Basic, VB .NET, Visual FoxPro, COBOL, Xbase++, dBase, PowerBASIC, etc.) capable of calling the Windows API. Win32 and Win64 DLLs are included.

When comparing the **SMTP/POP3/IMAP Email** component library against our competition, note that:

- SEE4D is a standard Windows DLL (NOT an OCX or ActiveX control) and is much smaller than a comparable OCX or ActiveX control.
- SEE4D does NOT depend on ActiveX or similar "support" libraries.
- The WIN32 version of SEE is fully thread safe.
- The SEE4D functions can be called from applications not capable of using controls.

MarshallSoft also has versions of the **SMTP/POP3/IMAP Email Engine** library for C/C++ (SEE4C), Visual Basic (SEE4VB), PowerBASIC (SEE4PB), Visual FoxPro (SEE4FP), Visual dBASE (SEE4DB), Xbase++ (SEE4XB) and COBOL (SEE4CB). All versions of the **SEE** library use the same DLLs (SEE32.DLL and SEE64.DLL). However, the examples provided for each version are written for the specified programming language.

The latest versions of **SMTP/POP3/IMAP Email Engine (SEE)** can be downloaded from our web site at

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

1.1 Features

Some of the many features of **SMTP/POP3/IMAP Email Engine** component library are as follows:

- SMTP client for sending email.
- POP3/IMAP client for receiving email.
- Send email with optional MIME or Quoted Printable attachments.
- Send email with inline embedded HTML, GIF, TIF, JPG, BMP and Rich Text attachments.
- Use with GMAIL/Yahoo/Hotmail servers requiring SSL with (free) STUNNEL proxy server.
- Get the number of messages on the POP3 email server.
- Get the header lines from any email on the POP3 email server, without reading the entire email.
- Remove contents of attachments before writing to disk.
- Delete any email on the POP3 server without reading it first.
- Copy any email on the POP3 server without deleting it.
- Check for the number of emails on the POP3 server.
- Receive any email on the POP3 server including MIME attachments.
- Forward email.
- Decode email from a file
- Run up to 64 independent WIN32 threads concurrently.
- Can send email to mail addresses on a distribution list.
- Supports SMTP (ESMTP) and POP3 authentication.
- Set return receipt; add TO, CC, BCC recipients
- Set minimum and maximum wait times for server response.
- Supports ISO-8859 (European character sets) and UTF-8 (16 bit character sets) messages.
- Can specify custom Content-Types; add custom header fields
- Includes over 65 functions for SMTP and POP3 mail control.
- Dozens of switches provided to control how email is sent or received.
- Supports setting priority via X-Priority header field.
- Start and terminate external programs from within an application.
- Can be used from GUI mode or console mode programs.
- Is fully thread safe.
- Implemented as a **standard** Windows DLL, which will work with all versions of Windows.
- Includes Win32 and Win64 DLLs.
- Supports all 32-bit and 64-bit versions of Borland /CodeGear Delphi, from Delphi 2 through Delphi 8, Delphi 2005 through Delphi 2010, Embarcadero Delphi XE through XE10 and Seattle.
- Does **not** depend on support libraries. Makes calls to core Windows API functions only.
- Can be used with any program (in any language) capable of calling Windows API functions such as Visual C++, .NET, Visual FoxPro, Visual Basic, VB.NET, Delphi, Xbase++, dBASE, COBOL, Access and Excel.
- Works with 32-bit and 64-bit Windows through Windows 11.
- License covers all programming languages.
- Free updates and technical support for one year.
- Royalty free distribution (no run-time fees) with your compiled application
- Evaluation versions are fully functional. No unlock code is required.

A good selection of Delphi example programs with full source code is included. Refer to Section 6 for more details on each of the example programs.

[PROGRAM]	[DESCRIPTION]
VERSION	Displays SEE Version/Build number and registration string. VERSION_PROJECT is similar but is for .NET.
CODETEST	Base64 encodes/decodes strings.
CONN	Tests connection to a specified server and port.
FORWARD	Forwards undecoded email.
FROM	Displays header information for email on server.
GB2312	Sends email that is GB2312 (simplified Chinese).
GETRAW	Downloads specified email without decoding.
HTML	Sends html encoded email with attachments.
ISO8859	Sends ISO-8859 encoded message and subject line
MAILER	Sends email with optional attachment. MAILER_PROJECT is similar but is for .NET.
MailSSL	Connects to server requiring SSL to send email.
Mparts	Sends multipart MIME email.
POP3RD	Specifies email message file to read and decode.
READER	Downloads email & attachments and saves to disk.
ReadSSL	Downloads email from POP3 server requiring SSL.
STATUS	Lists all email on server and displays DATE, FROM and SUBJECT header fields. STATUS_PROJECT is similar but for .NET.
ImapFlagsT	Tests manipulation of flaps on IMAP server. ImapFlgsT_Project is similar but is for .NET.
ImapSEARCH	Tests IMAP search capability. ImapSearch_Project is similar but is for .NET.

Also see EXAMPLES.TXT in the DOCS directory for a list of the examples provided for a particular compiler.

1.2 Documentation Set

The complete set of documentation consists of three manuals in Adobe PDF format. This is the first manual (SEE_4D) in the set.

- SEE4D Programmer's Manual (SEE_4D.PDF)
- SEE User's Manual (SEE_USR.PDF)
- SEE Reference Manual (SEE_REF.PDF)

The SEE_4D Programmer's Manual is the language specific (Delphi) manual. All language dependent programming issues such as compiling, compilers and example programs are discussed in this manual. Read this manual first.

The SEE User's Manual (SEE_USR) discusses email processing as well as language independent programming issues. Purchasing and licensing details are also provided.

The SEE Reference Manual (SEE_REF) contains details on each individual SEE function. A list of error codes is also provided.

Online documentation can be accessed on the **SMTP/POP3/IMAP Email Engine for Delphi** product page at:

http://www.marshallsoft.com/see4d.htm

1.3 Example Program

The following example demonstrates the use of some of the **SMTP/POP3/IMAP Email for Delphi** component library functions.

```
{connect then send email; assume seeAttach previously called}

function SendMail(ToList,Subject,Message: AnsiString):Integer;
var
  Code : Integer;
  SmtpHost : AnsiString;    {server name}
  SmtpUser : AnsiString;    {user name}
  SmtpPass : AnsiString;    {password}
  SmtpFrom : AnsiString;    {sender's email address}
  SmtpReply : AnsiString;   {reply email address}
  SmtpTo : AnsiString;      {recipient's email address}
  SmtpSubject : AnsiString; {email subject}
  SmtpMessage : AnsiString; {email message or file name of message}
  ccList : AnsiString;      {email CC list}
  bccList : AnsiString;     {email BCC list}
  Attachments : AnsiString; {list of attachment filenames}
begin
  {hard code server info}
  SmtpHost := 'smtp.my-isp.com' + Chr(0);
  SmtpUser := 'my-user-name' + Chr(0);
  SmtpPass := 'my-password' + Chr(0);
  SmtpFrom := '<mike@my-isp.com>' + Chr(0);
  SmtpReply:= '<mike@my-isp.com>' + Chr(0);
  ccList := Chr(0);
  bccList := Chr(0);
  Attachments := Chr(0);
  {specify the port to connect on (default port is 25)}
  seeIntegerParam(0, SEE_SMTP_PORT, 587);
  {enable "SMTP Authentication"}
  seeIntegerParam(0, SEE_ENABLE_ESMTP, 1);
  {specify the user name and password for SMTP authentication}
  seeStringParam(0, SEE_SET_USER, @SmtpUser[1]);
  seeStringParam(0, SEE_SET_SECRET, @SmtpPass[1]);
  {connect to SMTP server}
  Code := seeSmtpConnect(0,@SmtpHost[1], @SmtpFrom[1], @SmtpReply[1]);
  if Code < 0 then
    begin
      SendMail := Code;
      exit;
    end;
  {send email}
  SendMail := seeSendEmail(0, @SmtpTo[1], @ccList[1], @bccList[1],
              @SmtpSubject[1], @SmtpMessage[1], @Attachments[1]);
end;
{initialization}
begin
end.
```

In the example program above, **seeSmtpConnect** is called to connect to the SMTP mail host. The SMTP server host name and your email address are required, while the "Reply-To" entry is optional.

seeSendEmail is then called, passing the addressee lists. The primary addressee is provided in the "To List". The CC ("Carbon Copy") lists additional recipients, as does the BCC (Blind Carbon Copy) list. The subject contains the email subject line. The message text is next. If it starts with the '@' symbol, it is considered the name of the file containing the email message. Lastly, the filename of any ASCII or binary attachment is specified.

1.4 Installation

(1) Before installation of SEE4D, a 32-bit or 64-bit Delphi compiler (any version) should already be installed.

(2) Unzip SEE4D841.ZIP (evaluation version) or SEExxxxx.ZIP (purchased version where xxxxx is your Customer ID) using any Windows unzip program.

(3) Run the installation program, SETUP.EXE, which will install all SEE4D files including copying SEE32.DLL and SEE64.DLL to the Windows directory.

(4) For a quick start, load project file VER_PRJ.DPR

1.5 Uninstalling

To uninstall SEE4D, delete SEE32.DLL and SEE64.DLL from your \WINDOWS directory, then delete the **SEE** project directory created when SEE4D was installed.

1.6 Pricing

A developer license for the SMTP/POP3/IMAP Email Library can be purchased for $115 USD. Purchasing details can be found in the SEE User's Manual, Section 1.4, "How to Purchase", (/docs/see_usr.pdf or http:/www.marshallsoft.com/see_usr.pdf).

Also see INVOICE.TXT provided with the evaluation version or order directly on our web site at

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

Registration includes one year of technical support and free updates. Purchased SEE DLLs never expire.

1.7 Updates

When a developer license is purchased, the developer will be sent a registered DLL plus a license file (SEExxxx.LIC, where xxxx is your Customer ID). The license file can be used to update the registered DLL's for a period of one year 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 and receive technical support. The license can be updated for:

- $33 if the update is ordered within one year of the original purchase (or previous update).
- $55 is the update is ordered between 1 and 3 years of the original purchase (or previous update).
- $77 if the update is ordered after three years of the original purchase (or previous update).

The update price includes technical support for an additional year. Note that the registered DLLs, (SEE32. DLL and SEE64.DLL) never expire. Refer to the file UPDATES.TXT located in the /SEE4D/DOCS directory for more information.

2 Library Overview

The **SMTP/POP3/IMAP Email** component library has been tested on multiple computers running Windows 95/98/Me/XP/2003/2008/2012/Vista/Windows 7/Windows 8 and Windows NT/2000.

The SEE4D library has been tested and works with all versions of Borland (CodeGear) Delphi including Delphi 2 – Delphi 8, Delphi 2005 – Delphi 2010, Embarcadero Delphi XE through XE7, and Turbo Delphi.

The SETUP installation program will copy the SEE DLLs to the Windows directory and copies the SEE4D files to the directory specified (default \SEE4D). Four sub-directories are created, as follows:

```
DOCS – All documentation files
APPS – All example code
DLLS – All DLL's
SSL  - Proxy server files
```

2.1 Dynamic Link Libraries

The **SMTP/POP3/IMAP Email** component library uses a Win32 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 the traditional static library that is bound to each and every application that uses it at link time.

An important advantage that DLL's have over other "popular" library formats such as VBX or OCX is that DLL's are callable by all Windows applications. Since DLL's are the building blocks of the Windows Operating System, they will not be replaced by a "newer technology".

2.2 Keycode

The SEE32.DLL has a keycode encoded within it. The keycode is a 9 or 10 digit decimal number and will be found in the file KEYCODE.PAS. The keycode for the evaluation version is 0. The developer will receive a new keycode and a new SEE32.DLL after purchasing a license. The KEYCODE is passed to **seeAttach**.

If you get an error message (value -74) when calling **seeAttach**, it means that the keycode in your application does not match the keycode in the DLL. After registering, it is best to remove the evaluation version of the SEE32.DLL from the Windows search path or delete it.

2.3 Win32 STDCALL and DECLSPEC

SEE32 is written in ANSI C and is compiled using the _stdcall and _declspec keywords. This means that SEE4D uses the same calling conventions and file naming conventions as the Win32 API. In particular, function names are NOT decorated. There are neither leading underscores nor trailing "@size" strings added to function names.

Any Windows application program may call the SEE library provided that the proper declaration file is used.

2.4 Error Display

The error message text associated with **SEE** error codes can be displayed by calling **seeErrorText**. Each sample program contains examples of error processing.

Also see the file **seeErrors.txt** for a list of all Winsock and **SEE** error codes.

2.5 Adding SEE4D to a Project

If you are running Delphi for .NET, copy SEE32UC.PAS to the same directory as your application program. If you are running Delphi for Win32 (Delphi 2 and above), copy SEE32.PAS. Also copy the file KEYCODE.PAS to this same directory. You will find these files in the APPS directory (folder) created when you ran SETUP, usually C:\SEE4D\APPS.

Next, add a reference to the files copied above to your "uses" clause in your application program. For example,

```
uses
  see32, keycode, ...
```

You can leave 'keycode' out above if you put your numerical keycode value (found in `KEYCODE.PAS`) directly into the call to `seeAttach`. For example,

```
{pass the key code}
Code := seeAttach(1, 123456789)
```

Lastly, add a reference to `SEE32` to your project file (*.DPR). For example,

```
uses
  SEE32 in 'SEE32.PAS', ...
```

3 Compiler Issues

The **SMTP/POP3/IMAP Email Engine for Delphi** component library supports all versions of CodeGear (Borland) Delphi for Win32 as well as Delphi for .NET as follows:

- Borland Delphi 2, 3, 4, 5, 6, 7 and 8.
- Borland Delphi 2005 and Delphi 2006
- Borland Turbo Delphi
- CodeGear Delphi 2007 through 2010
- Embarcadero Delphi XE through XE10 (Win32 & Win64)

3.1 Delphi Versions

Applications written with Delphi link with the same identical DLL as for applications written in all other supported languages, such as C/C++ and Visual Basic.

3.1.1 Delphi 1

The first release of Borland Delphi (version 1) generated Win16 code.

SEE4D does not support 16-bit applications.

3.1.2 Delphi 2

Delphi version 2 and above generates Win32 code and link with SEE32DLL. Strings can be much larger than 255 bytes.

Delphi 2 seems to have a problem with some of the string functions. Although the default is "large strings", some of the string functions (such as StrPas) copy only 255 bytes. The MYSTRING.PAS unit contains a replacement unit to use instead of StrPCopy.

3.1.3 Delphi 3

Delphi 3 also has some problems with PChar string functions such as StrPCopy. See the previous section.

3.1.4 Delphi 4, 5, and 6.

There are no known Delphi problems impacting our example programs in Delphi version 4 and above.

3.1.5 Delphi 7 and 8.

Beginning in Delphi 7, the filename of a unit must match the unit name. Delphi 8 was a .NET only release.

3.1.6 Delphi 2005, Delphi 2006, and Delphi 2007

The Delphi 2005/2006/2007 compilers support both Win32 and the Microsoft .NET Framework.

When loading Delphi for Win32 projects with Delphi 2005/2006/2007, a window entitled "Project Upgrade" will be displayed:

This project must be upgraded before it can be opened. Please select which project type you wish to target:

```
( ) Delphi for .NET
( ) Delphi for Win32
```

Choose "Delphi for Win32" for all projects except "*.bdsproj" projects, which are Delphi for .NET projects.

3.1.7 Delphi 2009, Delphi 2010 and Delphi XE

In Delphi 2009 the definition of PChar was changed from a pointer to a 8-bit character to a pointer to a 16-bit character, also known as a “wide character”. For this reason PAnsiChar must be used rather than PChar for pointers to buffers that are passed to SEE functions. Refer to the Win32 example programs in the APPS directory.

3.1.8 Delphi XE2 through XE10

Delphi XE2 through XE10 can create both 32-bit and 64-bit executables. Note that although a 32-bit executable can run on both 32-bit and 64-bit Windows machines, a 64-bit executable can run only on a 64-bit Windows machine.

Step 1: Edit Files

1. Open file *_PGM.PAS using any text editor.
2. Replace SEE32 with SEE64 in the "uses" clause.
3. Open file *_PRJ.DPR using any text editor.
4. Replace SEE32 with SEE64.

Step 2: Delete Win32 files

1. Delete all *.DCU files that may have been created under Win32 Delphi.
2. Delete all *.DPROJ files that may have been created under Win32 Delphi.

Step 3: Change Win32 to Win64 in Delphi XE2/XE3/XE4 Project Manager

Start Delphi XE2, load the project *_PRJ.DPR, then open the Delphi Project Manager.

1. Right click on "Target Platforms (Win32)" and click on "Add Platform...".
2. When the "Select Platform" window is displayed, click "64-bit Windows".
3. Click [+] box to left of "Target Platforms (Win32)"
4. Right click on "32-bit Windows", then click "Remove Platform"

3.2 Calling SEE functions From Delphi

SEE functions have just two types of arguments: Integers and pointers to strings. For example,

```
function seeCommand(Chan:Integer; Text:PAnsiChar):Integer; stdcall;
```

as declared in both SEE32.PAS and SEE64.PAS.

For example, suppose that we want to send the "no operation" command ("NOOP") to the SMTP server once we are connected. The code segment could look like

```
var
  Code : Integer;
  Text : AnsiString;
begin
  ...
  Text := 'NOOP' + Chr(0);
  Code := seeCommand(0, @Text[1]);
  ...
```

Strings are declared as AnsiString. All strings passed to SEE functions should be null terminated. The "@Text[1]" argument above passes the address of the first byte of the string to seeCommand.

3.3 Delphi Personalities

Beginning with Delphi 2003, Delphi has two “personalities”: (1) Win32 Delphi and (2) Delphi for .NET. Win32 Delphi programs are a continuation of the Delphi language as seen in earlier versions of Delphi. Delphi .NET is a version of Delphi designed to use the Microsoft .NET Framework.

3.4 Compiling Programs

The example programs are compiled from the Delphi development environment using the provided Delphi project files (*.DPR).

Refer to Section 4.0 "Example Programs" for more details on each of the example programs.

4 Example Programs

SMTP programs send email, while POP3/IMAP programs check and download mail.

Each of the following example programs uses the "display" unit and the "SEE" unit:

```
DISPLAY.PAS   : Display unit source code.
SEE.PAS       : SEE Unit source code.
```

The DISPLAY.PAS unit is used to display text in Delphi memos. DISPLAY.PAS contains 4 procedures:

```
DisplayChar   : Displays character.
DisplayString : Displays string.
DisplayLine   : Displays line.
DisplayError  : Displays error message.
```

Delphi for .NET projects must be compiled with Delphi for .NET 2005/2006/2007. See section 3.1.

4.1 Connectionless Example Programs

Several example programs do not require a connection to a server.

4.1.1 VERSION

This simple program displays the **SEE** version number, build number, and registration string taken from SEE32.DLL. Its purpose is display the **SEE** version, build, and registration string as well as to verify that SEE32.DLL is being found and loaded by Windows The VERSION program does not attempt to connect to any server.

The Delphi for Win32 VERSION project files are:

```
VER_PRJ.DPR  : Delphi project file.
VER_PGM.DFM  : Delphi form file.
VER_PGM.PAS  : Program source code.
```

4.1.3 VERSION_PROJECT

This is the Delphi for .NET equivalent of the Delphi for Win32 VERSION program.

The Delphi for .NET project files are:

```
Version_Project.bdsproj        : Delphi for .NET project file
Version_Project.dpr            : Delphi for .NET project file
Version_WinForm.pas            : Delphi for .NET form source
Version_WinForm.resx           : Delphi for .NET resource file
Version_WinForm.TWinForm.resources  : Delphi for .NET resource file
```

4.1.2 CODETEST

The CODETEST example program demonstrates how to use **seeEncodeBuffer** and **seeDecodeBuffer**, which BASE64 encodes and decodes several test strings. The CODETEST example program also demonstrates the use of **seeEncodeUTF8** and **seeDecodeUTF8**.

The Delphi for Win32 CODETEST project files are:

```
CODE_PRJ.DPR  : Delphi project file.
CODE_PGM.DFM  : Delphi form file.
CODE_PGM.PAS  : Program source code.
```

4.1.3 VERSION_PROJECT

This is the Delphi for .NET equivalent of the Delphi for Win32 VERSION program.

The Delphi for .NET project files are:

```
Version_Project.bdsproj        : Delphi for .NET project file
Version_Project.dpr            : Delphi for .NET project file
Version_WinForm.pas            : Delphi for .NET form source
Version_WinForm.resx           : Delphi for .NET resource file
Version_WinForm.TWinForm.resources  : Delphi for .NET resource file
```

4.1.4 CONN

The CONN example program tests if a SMTP, POP3, or IMAP server is accepting connections on a specified port. This is very useful when attempting to connect to a new email server.

The user name and password are not used in order to connect to a server, but rather are used after the connection has been accepted by the server.

The Delphi for Win32 CODETEST project files are:

```
CONN_PRJ.DPR  : Delphi project file.
CONN_PGM.DFM  : Delphi form file.
CONN_PGM.PAS  : Program source code.
```

4.2 SMTP Example Programs

There are nine SMTP email example programs. SMTP programs send email using an SMTP server.

4.2.1 FORWARD

The FORWARD example program forwards an email message to a new recipient. Only undecoded email messages can be forwarded.

Undecoded email message can be downloaded using the GETRAW and READER example programs.

The Delphi for Win32 FORWARD project files are:

```
FWD_PRJ.DPR  : Delphi project file.
FWD_PGM.DFM  : Delphi form file.
FWD_PGM.PAS  : Program source code.
```

4.2.2 HELLO

The HELLO program emails a short message. Before compiling, HELO_PGM.PAS must be edited (about line 43) with your email parameters. HELLO uses the SEE wrapper unit (SEEW.PAS).

The HELLO project files are:

```
HELO_PRJ.DPR  : Delphi project file.
HELO_PGM.DFM  : Delphi form file.
HELO_PGM.PAS  : Program source code.
```

Compare Delphi for Win32 HELLO with the MAILER example program.

4.2.3 HTML

The HTML example program connects to an SMTP server and emails an HTML encoded file (HTML.HTM) containing inline embedded graphics (IMAGE1.GIF and IMAGE2.GIF) as well as an alternate text file (HTML.TXT).

Note that the defaults are read from the file HTML.INI at runtime.

The Delphi for Win32 HTML project files are:

```
HTML_PRJ.DPR  : Delphi project file.
HTML_PGM.DFM  : Delphi form file.
HTML_PGM.PAS  : Program source code.
```

4.2.4 ISO 8859

The ISO 8859 example program sends a text message and subject line that are ISO-8859 encoded. The recipient's email client will be able to display the email message using the specified ISO character set provided that it is capable of identifying ISO-8859 MIME parts (such as MS OutLook).

The Delphi for Win32 ISO 8859 project files are:

```
ISO_PRJ.DPR  : Delphi project file.
ISO_PGM.DFM  : Delphi form file.
ISO_PGM.PAS  : Program source code.
```

4.2.5 GB 2312

The GB 2312 example program sends a text message that is GB2312 (simplified Chinese) encoded. The recipient's email client will be able to display the email message using the specified GB2312 character set provided that it is capable of identifying GB2312 MIME parts (such as MS OutLook).

The Delphi for Win32 GB 2312 project files are:

```
GB_PRJ.DPR  : Delphi project file.
GB_PGM.DFM  : Delphi form file.
GB_PGM.PAS  : Program source code.
```

4.2.6 MailSSL

The MailSSL example program emails a specified email message connecting to a SMTP server that requires SSL, such as Gmail, Hotmail, and Yahoo. Be sure to read the section "Using Stunnel" in the SEE User's Manual (/DOCS/SEE_USR.PDF or www.marshallsoft.com/see_usr.pdf) in the DOCS directory.

The Delphi for Win32 MailSSL project files are:

```
MailSSL_PRJ.DPR  : Delphi project file.
MailSSL_PGM.DFM  : Delphi form file.
MailSSL_PGM.PAS  : Program source code.
```

4.2.7 MAILER

The MAILER example program emails a message, including optional MIME attachments.

The Delphi for Win32 MAILER project files are:

```
MAIL_PRJ.DPR  : Delphi project file.
MAIL_PGM.DFM  : Delphi form file.
MAIL_PGM.PAS  : Program source code.
```

Note that the MAILER program uses the "indirect" method (refer to the SMTP/POP3/IMAP User's Manual, Section_6.1, "Indirect Method" (/DOCS/SEE_USR.PDF or online at (www.marshallsoft.com/see_usr.pdf).

The function **seeDriver** is called automatically. An hourglass cursor is displayed while your email is being sent to the server.

4.2.8 MAILER_PROJECT

This is the Delphi 2005/2006/2007 NET equivalent of the Delphi for Win32 MAILER program above.

The Delphi for .NET project files are:

```
Mailer_Project.bdsproj        : Delphi for .NET project file
Mailer_Project.dpr            : Delphi for .NET project file
Mailer_WinForm.pas            : Delphi for .NET form source
Mailer_WinForm.resx           : Delphi for .NET resource file
Mailer_WinForm.TWinForm.resources  : Delphi for .NET resource file
```

4.2.9 PART

The PART example program sends a multipart MIME email in which the programmer specifies the Content-Type headers for each attachment.

The two attachment types specified in this example are a sound file (*.wav) and of PDF file (*.pdf).

The Delphi for Win32 PART project files are:

```
PART_PRJ.DPR  : Delphi project file.
PART_PGM.DFM  : Delphi form file.
PART_PGM.PAS  : Program source code.
```

4.3 POP3/IMAP Example Programs

There are six POP3 email example programs that read email using the POP3 server or IMAP server.

4.3.1 AUTO

AUTO ("auto-responder") uses two channels concurrently to read email on the POP3 server and at the same time automatically respond to all new email using the SMTP server. Edit the TCP/IP parameters in AUTO_PGM.PAS (line 82) before compiling. AUTO uses the SEEW.PAS wrapper unit.

The Delphi for Win32 AUTO project files are:

```
AUTO_PRJ.DPR  : Delphi project file.
AUTO_PGM.DFM  : Delphi form file.
AUTO_PGM.PAS  : Program source code.
```

4.3.2 GET RAW (GRAW)

GETRAW is an example program that downloads a specified email message without decoding it (in "raw" format). This is used to see what the email looks like on the server. Also see the READER example program, which can also download email without decoding it.

All required parameters are taken from the dialog box at runtime.

The Delphi for Win32 project files are:

```
GRAW_PRJ.DPR  : Delphi project file.
GRAW_PGM.DFM  : Delphi form file.
GRAW_PGM.PAS  : Program source code.
```

4.3.3 POP3READ

The Pop3Read example program uses the **seePop3Source** function to specify an (undecoded) email message file to be read and decoded.

The Delphi for Win32 Pop3Read project files are:

```
POP3_PRJ.DPR  : Delphi project file.
POP3_PGM.DFM  : Delphi form file.
POP3_PGM.PAS  : Program source code.
```

4.3.4 READER

READER can read email, including multiple MIME attachments, from your POP3 server, optionally deleting each email after being read.

The Delphi for Win32 READER project files are:

```
READ_PRJ.DPR  : Delphi project file.
READ_PGM.DFM  : Delphi form file.
READ_PGM.PAS  : Program source code.
```

Note that the READER program uses the "direct method" (refer to SMTP/POP3/IMAP User's Manual SEE_USR). The **seeDriver** is explicitly called. The progress of the incoming email is displayed.

The READER program can also download email without decoding. See the note "PROGRAMMER" in READ_PGM.PAS

4.3.5 ReadSSL

The ReadSSL example program downloads email messages from a POP3 server that requires SSL, such as Gmail, Hotmail, and Yahoo. Be sure to read the section "Using Stunnel" in the SEE User's Manual (/DOCS/SEE_USR.PDF or online at (www.marshallsoft.com/see_usr.pdf).

The Delphi for Win32 ReadSSL project files are:

```
ReadSSL_PRJ.DPR  : Delphi project file.
ReadSSL_PGM.DFM  : Delphi form file.
ReadSSL_PGM.PAS  : Program source code.
```

4.3.6 STATUS

STATUS reads the number of email messages waiting on a POP3 server, and displays the "DATE:", "FROM:", and "SUBJECT:" header fields from each email.

The Delphi for Win32 STATUS project files are:

```
STAT_PRJ.DPR  : Delphi project file.
STAT_PGM.DFM  : Delphi form file.
STAT_PGM.PAS  : Program source code.
```

4.3.7 STATUS_PROJECT

This is the Delphi 2005/2006/2007 NET equivalent of the Delphi for Win32 STATUS project above.

The Delphi for .NET project files are:

```
Status_Project.bdsproj        : Delphi for .NET project file
Status_Project.dpr            : Delphi for .NET project file
Status_WinForm.pas            : Delphi for .NET form source
Status_WinForm.resx           : Delphi for .NET resource file
Status_WinForm.TWinForm.resources  : Delphi for .NET resource file
```

4.4 IMAP-Only Email Example Programs

There are four IMAP-only email example programs. These examples access the IMAP server.

4.4.1 ImapFlagsT

The ImapFlagsT example program tests the manipulation of flags on the IMAP server. It reads, sets, and deletes certain flags for the specified email message on the IMAP server.

IMAP flags are:

```
\Seen      Message has been read
\Answered  Message has been answered
\Flagged   Message is "flagged" for urgent/special attention
\Deleted   Message is "deleted" for removal by later EXPUNGE
\Draft     Message has not completed composition (marked as a draft).
\Recent    Message has arrived since the previous time this mailbox was
           selected. ["\Recent" may be fetched but not stored]
```

The Delphi for Win32 project files are:

```
ImapFlagsT_PRJ.DPR  : Delphi project file.
ImapFlagsT_PGM.DFM  : Delphi form file.
ImapFlagsT_PGM.PAS  : Program source code.
```

4.4.2 ImapFlagsT_Project

The ImapFlagsT_Project example program is the Delphi for .NET equivalent of the Delphi for Win32 ImapFlagsT project.

The Delphi for .NET project files are:

```
ImapFlagsT_Project.bdsproj        : Delphi for .NET project file
ImapFlagsT_Project.dpr            : Delphi for .NET project file
ImapFlagsT_WinForm.pas            : Delphi for .NET form source
ImapFlagsT_WinForm.resx           : Delphi for .NET resource file
ImapFlagsT_WinForm.TWinForm.resources  : Delphi for .NET resource file
```

4.4.3 ImapSearch

The ImapSearch example program tests IMAP search capability.

See ImapSearch.txt or http://www.marshallsoft.com/ImapSearch.htm
for a complete list of all IMAP search strings.

Example search strings as passed to **seeImapSearch**():

```
SEEN
SEEN NOT ANSWERED
FLAGGED SINCE 1-Feb-2008 NOT FROM "Smith"
LARGER 10000 NOT SEEN
```

The Delphi for Win32 project files are:

```
ImapSearch_PRJ.DPR  : Delphi project file.
ImapSearch_PGM.DFM  : Delphi form file.
ImapSearch_PGM.PAS  : Program source code.
```

4.4.4 ImapSearch_Project

The ImapSearch_Project example program is the Delphi for .NET equivalent of the Delphi for Win32 ImapSearch project.:

The Delphi for .NET project files are:

```
ImapSearch_Project.bdsproj        : Delphi for .NET project file
ImapSearch_Project.dpr            : Delphi for .NET project file
ImapSearch_WinForm.pas            : Delphi for .NET form source
ImapSearch_WinForm.resx           : Delphi for .NET resource file
ImapSearch_WinForm.TWinForm.resources  : Delphi for .NET resource file
```

5 Revision History

The SMTP/POP3/IMAP Email Engine DLL (SEE32.DLL) is written in ANSI C. All programming language versions of SEE (C/C++, .NET, Visual Basic, VB .NET, PowerBASIC, Visual FoxPro, Delphi, dBase, Xbase++, COBOL, and Fortran) use the same SEE32.DLL.

Version 1.0: June 22, 1998

- Initial release.

Version 2.0: September 28, 1998.

- A major update adding POP3 capability.
- Another SMTP example program (BCAST).

Version 2.1: November 28, 1998.

- Time zone calculated automatically.
- Fixed bug in seeClose.
- Corrected POP3 problem when boundary definition on 2nd line.
- Added support for alternate MIME boundaries.
- Added seeVerifyUser function.
- Added SEE_GET_REGISTRATION, SEE_GET_CONNECT_STATUS,
- SEE_GET_ATTACH_COUNT, and SEE_GET_LAST_RESPONSE.
- Added GETDOC, SEEVER, and VERUSR example programs.
- SMTP performance improved.
- Added seeEncodeBuffer and seeDecodeBuffer functions.
- Added SEE_FILE_PREFIX and SEE_SET_REPLY parameters.

Version 3.0: April 12, 1999.

- Modified SEE to be fully threadable (adding seeAttach and seeRelease).
- Added seeGetEmailUID function.
- Handles "inline" email text properly.
- Optionally decodes unnamed attachments.
- Added ability to add header lines (SEE_SET_HEADER).
- Can use alternate ports for SMTP or POP3.
- . Quoted-printable messages can handle soft line breaks.
- Quoted-printable message can handle embedded HTML text.
- Delphi wrapper unit "SEEW" added.

Version 3.1: July 16, 1999.

- Support ISO-8859-1 (Q or B) encoded attachment filenames.
- Support ISO-8859-1 (Q only) on subject line
- SEE_SAVED_TO_MSG added.
- "+OK" line not written to email message file.
- Added seeExtractLine function.
- Added REGME example program.

Version 3.2: February 7, 2000.

- Can decode printed quotable attachments!
- seeGetEmailLines can use internal memory.
- Added SEE_WRITE_TO_LOG to seeStringParam.
- Added SEE_GET_ATTACH_NAMES to seeDebug to get attachment filename list.
- Ability to reset the SEE_SET_HEADER header string to "nothing".
- Improvements in dynamic memory usage.
- Added GETRAW and AUTO examples.
- Added seeCommand function.

Version 3.3: October 3, 2000

- seeGetEmailLines can use internal memory.
- Added SEE_COPY_BUFFER [seeDebug] to copy internal buffer.
- Added SEE_WRITE_TO_LOG [seeStringParam] to allow user to write to LOG file.
- Added SEE_GET_ATTACH_NAMES [seeDebug] to get attachment filename list.
- Ability to reset the SEE_SET_HEADER [seeStringParam] to "nothing".
- Added seeCommand function.
- Allow TIC marks (0x27) in VerifyAddressChars().
- Added SEE_GET_LAST_RECIPIENT to seeDebug.
- Added seconds to date string on outgoing email.
- Attachment name is saved when attachment file is closed.
- Added SEE_PATH_DELIMITER to seeIntegerParam().
- Added seeAbort function.
- VerifyFormat rejects "@domain" and "name@" addresses.
- Added "SEE_SET_FROM" so can change "From:" header at runtime.
- Delimiters (CR/LF) sent with command in one network transmission [seeWriteLine].
- Added QUOTED_USER, SEE_SET_CONTENT_TYPE, and SEE_SET_TRANSFER_ENCODING.
- Added SEE_ATTACH_DELIMITER and ability to specify different attachment filename in email.
- Added SEE_ADD_HEADER to seeStringParam.
- Added SEE_WRITE_BUFFER to seeDebug (see seeGetEmailLines)
- Added SEE_ENABLE_GIF to send GIF images inside email.

Version 3.4: July 24, 2001

- Supports "AUTH LOGIN" and "AUTH CRAM-MD5" (SMTP) authentication.
- SmtpResponse accepts response line without message.
- Supports ISO-8859-1 (base-64) encoding on subject line.
- Supports "APOP" authentication (POP3 servers).

Version 3.5: March 5, 2002

- Added support for "AUTH PLAIN".
- Recognize multiple AUTH methods on one line, such as "AUTH PLAIN LOGIN CRAM-MD5".
- Added SEE_FORCE_INLINE -- attachments are inline text rather than base64 encoded.
- Added SEE_SET_ATTACH_CONTENT_TYPE -- user can specify content type for attachments.
- Added SEE_ATTACH_BASE_NUMBER -- attachments named "1.att", "2.att", etc.
- Don't close socket (seeClose) if socket is already closed.
- NBR_CHANS set to 128 for Win32.
- SEE_RAW_MODE reads complete lines rather than buffers.
- Added seeQuoteBuffer() -- used to prepare ISO-8859 headers.
- Will continue with sending DATA (rather than return error) if have at least one recipient.
- Call seeStatistics(Chan, SEE_GET_LAST_RECIPIENT) to get # recipients accepted by server.
- Added "Broadcast" (BCST_) and "Codetest" (CODE_) example programs.
- Added SEE_IGNORE_REJECTED to ignore error returned if recipient is rejected.

Version 3.6: March 20, 2003

- Added seeSendHTML() function.
- Looks for multipart/related as well as multipart/alternative message parts.
- Added SEE_HTML_CHARSET (CHARSET_US and CHARSET_8859)
- Generic multipart boundary definitions handled (not just alternate, related, ...)
- CR/LFs preserved in multiline "Subjects:" headers.
- Handle case where "MIME-Version: 1.0" statement does not proceed all other MIME statements
- MAX_BUF increased from 2048 to 8192 for WIN32
- Virtual socket # written to log file when created (vsGetSocket) & released (vsCloseSocket).
- Write to email file if "MIME-Version" was not seen.
- vSock released properly in seeClose.
- Terminating ALT boundary not written if HTML file is passed from memory (not a file)
- Alternate text in seeSendHTML can be file (if prefixed with '@')
- Added seeEncodeUTF8 and seeDecodeUTF8 functions.
- Delimiters separating email addresses and pathnames changed to a semicolon.
- Added ISO_8859, WIN_1252, and WIN_1255 character set types.

Version 3.7: January 28, 2005.

- Terminating ALT boundary not written if HTML file is passed from memory (not a file).
- Alternate text in seeSendHTML can be file (if prefixed with '@')
- Added seeEncodeUTF8 and seeDecodeUTF8 functions
- AddrDelimiter and PathDelimiter changed to ';' (semicolon)
- Added QUOTED_WIN_1252 and QUOTED_WIN_1255.
- User headers written even if no subject
- Corrected problem: User Content-Type wasn't being sent if no quoting
- Added SEE_HIDE_HEADERS -- overrides any conflicting flags
- Fixed problem with "Filename=" extraction.
- Replaced OF_READ|OF_SHARE_DENY_WRITE with OF_SHARE_DENY_WRITE in _lopen
- Filename added to SEE_CANNOT_CREATE & SEE_CANNOT_OPEN error messages.
- Multi-line subject headers supported in seeGetEmailFile.
- ReadMsgLine uses Allow8Bits to decide if it should quote or not
- Added SEE_SET_DEFAULT_ZONE
- Increased buffer size for challenge string in authenticated SMTP connections.
- Added WriteToLog(), WriteClientTempToLog(), and WriteToLastLog() to centralize log writing.
- Nulls are replaced by spaces in all incoming data.
- Added support for "=?US-ASCII?B?" encoded filenames
- Fixed problem quoting line starting with '.' and having non-ASCII characters.
- Fixed SMTP problem when attaching large number of files (seeWriteSocket,seeWriteLine,seeWriteString).
- Added IgnoreErrorStatus (default TRUE) that skips socket error check in STATE_CONNECT
- Fixed problem with Content-Type prefix (set by SEE_WRITE_CONTENT_TYPE).
- Scan subjects & filenames for "big5" encoding like iso-8859
- Only one of TO, CC, and BCC must contain a recipient.
- Maximum text line length default increased to 1000.
- Added SEE_REPLACE_WITH_COMMAS to override replacement of delimiters with commas.
- SEE_FILE_PREFIX parameters set base for attachment file prefixes.
- Added seeAttachmentParams function.

Version 4.0: August 17, 2006.

- Always an error if "relay", "gateway", or "not local" is in the text of the server's response, regardless of SEE_IGNORE_REJECTED.
- Forwarded header lines written to message/rfc822 (attachment) file.
- Each POP3 message optionally saved to disk in raw (undecoded) format in seeGetEmailFile.
- Added function seeForwardEmail().
- Added function seePop3Source().
- Maximum internal buffer size increased from 8 KB to 16 KB.
- Alternate boundaries w/o enclosing quotes are supported.
- FORWARD and Pop3Read example programs added.
- Added function seeByteToShort
- Added function seeShortToByte

Version 5.0: May 14, 2008 (Win32 Version only)

- Added seeSetErrorText.c example program
- Added LoadLib.c example program.
- Added IMAP capability. IMAP-only functions are:
  1. seeImapConnect : Connect to IMAP server.
  2. seeImapFlags : Get, set, or delete message flags.
  3. seeImapSearch : Search for messages with specified flags.
  4. seeImapMsgNumber : Gets message numbers from buffer filled by seeImapSearch.
  5. seeImapSelectMB : Selects IMAP mailbox.
  6. seeImapDeleteMB : Delete a mailbox.
  7. seeImapCreateMB : Create a new mailbox.
  8. seeImapRenameMB : Rename mailboxes.
  9. seeImapCopyMBmail : Copy messages from selected mailbox to specified mailbox.
  10. seeImapListMB : List all available mailboxes.
- Added ImapFlagsT, ImapSearch, and ImapMBtest example programs.
- Pass NULL for filename to seePop3Source / seeImapSource to revert back to server processing.

Version 5.1: May 13, 2009 (Win32 and Win64) versions

- Fixed code for IMAP_SEARCH_MSG_COUNT in seeImapMsgNumber
- Appended CR/LF to text returned by seeGetEmailUID
- Fixed problem with STATE_POP3_DELETE (call exiting via STATE_POP3_DELETE_OK)
- Consider TAB's as ASCII characters (TABIsASCII=1)
- Added EnableHeaders to enable/disable writing of headers.
- Don't write blank line after headers (in STATE_SMTP_BODY) if EnableHeaders = 0
- Write the # bytes written to mail file in the log file.
- Never write boundaries to the email file.
- Fixed bug: seeGetEmailCount works with all IMAP mailboxes (not just InBox)
- Added seeStartProgram and seeKillProgram to start/terminate external programs.
- Fixed problem with blocking mode so connect timeout works.
- Added seeSmtpTarget that writes SMTP output to a file.
- Fixed problem with seeSendEmail (w/ attachment) after forwarding email.
- Added Win64 DLL (not yet supported by Delphi).

Version 5.2: April 15, 2010. (Win32 and Win64) versions.

- Added seeSleep function (for languages not having a native Sleep call).
- The HELO command passes the computer name rather than its IP address.
- Bug Fix: All handles closed before memory blocks are freed.
- Bug Fix: Multiline "To:" header preserved in incoming email.
- Bug Fix: seeSmtpTarget now always closes files.
- Bug Fix: seePop3Source now always closes files.
- Bug Fix: Multiple IMAP response lines now handled properly by seeCommand.
- Added UTF8 character set support (CHARSET_UTF8).
- Added check for "MX lookup failure" when reading incoming mail.
- Added check for "Invalid MX record" when reading incoming mail.
- Changed IMAP list command argument default from ~/* to "" "*".
- Added SEE_SET_IMAP_LIST_ARG to seeStringParam (sets IMAP list command argument)
- Added seeReadQuoted function: reads a file and quotes the contents as it writes to a buffer.
- Added "Buffer overflow" error code.
- Added QUOTED_ISO_8859_2 to seeIntegerParam for sending ISO_8859_2 encoded emails.
- Added QUOTED_ISO_8859_7 to seeIntegerParam for sending ISO_8859_7 encoded emails.
- Added SEE_GUT_ATTACHMENTS to seeIntegerParam to remove contents of incoming attachments.

Version 6.0: February 15, 2011 (Win32 and Win64) versions.

- Better integration to the Stunnel proxy server.
- Added seeSmtpConnectSSL and seePop3ConnectSSL.
- Added seeIsConnected.
- Fixed: Can now have leading period in alternate text.
- Added SEE_SET_LOCAL_IP (seeStringParam) to specify local IP.
- Added CHARSET_WIN_1250.
- Changed (default) MaxResponseWait from 10 secs to 25 secs.
- Added SEE_SET_HELO_STRING.
- Fixed problem with reading POP3 from file.
- Add support for ISO-8859-3 and ISO-8859-4.

Version 7.0: November 21, 2011

- Added support for 64-bit Delphi XE2
- Fixed problem decoding some "ISO-8859" subjects
- Fixed problem with wrong content type when using seePop3Rd
- Fixed problem with seeAttachmentParams
- Added seeImapConnectSSL()
- ParseISO removes iso-8859-15 encoding from incoming Subject, etc.
- "To:" and "CC:" strings decoded (base64 & quoted)
- Decode quoted UTF-8 subject strings
- Replace underscore with blank (RFC2047) in UnQuote
- Added ".png" to image types
- Call seeStringParam(Chan, SEE_SET_HELO_STRING, '*') to use machine name for HELO string
- Call seeStringParam(Chan, SEE_LOG_FILE, "\0") to disable logging
- Recognizes iso-2022-jp
- Added seeSetProxySSL()
- Modified seeSmtpConnectSSL(), seePop3ConnectSSL(), seeImapConnectSSL(). Includes changes so that Stunnel (used for email services requiring SSL) is automatically configured, loaded, and unloaded without any user intervention.
- Use large buffer (64K) for IMAP server response on channel 0

Version 7.1: April 16, 2012

- Can pass full pathname for ProxyEXE and ProxyCert in seeSetProxySSL.
- Buffer sizes for ProxyEXE & ProxyCert (seeSetProxySSL) increased from 64 to 256 chars.
- (NOTE: can an no longer pass a null string for PEM certificate)
- seeRelease() kills all running copies of Stunnel started by SEE.
- Password characters not written to log file (PASS ****) & AUTH transmissions
- Added SEE_SET_CONNECT_ATTEMPTS that sets max connection attempts (1 to 12)
- Fixed problem: ImapConnect not returning error if bad login.
- SEE closes all process handles for all external program started by SEE.

Version 7.2: October 1, 2013

- Added support for Delphi XE3 and XE4
- Increased the maximum number of channels from 32 to 64.
- Allow multiple subject lines in incoming email.
- Added SEE_REPLACE_UNDERSCORES to seeIntegerParam() to disable replacement of underscores with spaces (RFC2047).
- Fixed problem with GMAIL IMAP connection.
- Can now decode Win1255 subjects.
- seeAbort now always closes attachment files.
- Fixed zone calculation for "half-zones".
- Added debug info to seeGetEmailCount().
- Added STUNNEL_DISABLE_LOGGING flag to seeSetProxySSL() that disables Stunnel logging.
- Fixed problem with SEE_ADD_HEADER when re-opening connection.
- Allow attachment filename to have a leading space.
- Added seeGetHeader() function with parameters SEE_GET_SUBJECT, SEE_GET_FROM, SEE_GET_REPLT_TO, SEE_GET_TO, and SEE_GET_DATE

Version 7.3: December 18, 2014

- Decodes UTF8 encoded attachment filenames.
- Diagnostics written to log file if missing '<' or '>' delimiters in email addresses.
- Added SEE_ALLOW_PARTIAL to seeIntegerParam which allows PARTIAL commands in IMAP.
- Added SEE_GET_UIDVALIDITY to seeStatistics which returns UID Validity in IMAP.
- Fixed problem with boundary buffer [64-bit only].
- Added seeConfigSSL() function which adds lines to the SSL configuration file.
- Added seeUnquote() function that unquotes quoted buffers.
- Added UTF8 quoting : seeIntegerParam(Chan, SEE_QUOTED_PRINTABLE, QUOTED_UTF8)

Version 7.4: April 29, 2016

- Changed: seeImapConnect() & seeImapConnect() now hide LOGIN password .
- Added: Content-Type marked automatically for PDF and WAV files.
- Fixed: socket forced closed if cannot connect to server.
- Fixed: replace non ASCII characters in the subject and header strings with the '_' character.
- Added: allow commas to be used in a filename itself (seeTestFileSet).
- Added: seeMakeSubject() to make ISO & UTF-8 quoted subject strings.
- Added: more diagnostics to the SEE log file.
- Added: new example program CONN that tests connection to server.

Version 7.5: September 25, 2017

- Fixed : Problem fixed in which two user headers can't be set.
- Added : SEE_SET_RCPT_TRACE_FILE added to seeStringParam to write RCPT trace to disk
- Added : Added seeSetCertAuth() to specify Certification Authority certs (for SSL)
- Added : Current directory filename is written to log file
- Added : The date stamp filename is written to the log file
- Added : The SEE version written to log file moved to just after log file is opened
- Added: Writes SSL file date stamps to SEE log file.

Version 8.0: October 30, 2018

- Fixed: Multiple subject lines were not always correctly combined (seeGetEmailFile).
- Fixed: Problem with SSL PLAIN authentication protocol corrected.
- Change: Base64 strings passed to seeDecodeBuffer() no longer required to end with CRLF.
- Change: Increased value of MaxConnectAttempts from 12 to 20.
- Added: Constant SEE_SHOW_PASSWORDS added that shows all passwords in log file.
- Added: Added function seeEncodeUTF8String() that encodes UTF8 strings.
- Added: Added function seeDecodeUTF8String() that decodes UTF8 strings.
- Added: Added constant SEE_GET_CUSTOMER_ID to function seeStatistics().
- Added: Added error code constant SEE_BAD_UTF8_CHAR.
- Added: Customer ID written to Stunnel configuration file.
- Added: Additional connection statistics written to SEE log file.
- Added: SEE writes SSL file date stamps to SEE log file.

Version 8.1: April 16, 2020

- Fixed: Fixed seeTestFileSet() problem (file critical section not initialized)
- Added: Added Windows error text written to log file when Windows returns an error
- Change: Files opened for read now in shared access (FILE_SHARE_READ)
- Fixed: Fixed "CID=" string in log file
- Added: IMAP response "+OK" accepted in addition to "OK"
- Fixed: Fixed problem with UnQuote (did not handle lower case hex correctly)
- Added: Added error code SEE_INDEX_RANGE
- Added: Added function seeSetFileBuffer() - used to pass buffer (not file) as attachment
- Added: Added BASE64_UTF8 to seeMakeSubject()
- Added: seeStartProgram() sets last error for subsequent call to seeDebug(Chan, SEE_GET_LAST_ERROR,...)
- Added: Added SEE_GET_LAST_ERROR (36) to seeDebug
- Added: Error text written to log file if Stunnel can't be started
- Fixed: Handles decoding ISO & UTF8 multi-line subjects correctly
- Added: Writes KeyCode value to log file if seeDebug(...,SEE_GET_REGISTRATION,...) is called
- Change: Change SEE_REPLACE_UNDERSCORES (EnableRFC2047) default to FALSE
- Added: Added SEE_GET_KEYCODE to seeStatistics()
- Change: Multi-line "Subject:" lines saved as lines (not concatenated) for retrieval by seeGetHeader()
- Change: Multi-line "To:" lines saved as lines (not concatenated) for retrieval by seeGetHeader()

Version 8.2: January 15, 2021

- Added: IMAP processing now traps "NO" and "BAD" responses during login.
- Change: Removed limits on the number & size of saved subject lines that can be returned by seeGetHeader()
- Change: Allows ISO & UTF subjects to be broken in mid-line.
- Change: SEE_REPLACE_UNDERSCORES (EnableRFC2047) default changed to TRUE
- Added: Added support for OAUTH2 - user MUST provide access token !
- Fixed: Base64 encoded passwords now replaced with astericks in log file (default)
- Change: Increased password buffer to 256 characters (supports SendGrid passwords)

Version 8.3: April 18, 2022

- Fixed problem when renaming attachments "on the fly" [oldname|newname].
- Added renaming of inline images "on the fly" [oldname|newname].
- Write authentication protocol selected by user to the log file.
- Added "Unexpected empty string" (SEE_EMPTY_STRING) error code.
- Allow "From:" header to be split over multiple lines.
- Protocol XOAUTH2 must be explicitly enabled - seeIntegerParam(0, SEE_AUTHENTICATE_PROTOCOL, AUTHENTICATE_XOAUTH2).

Version 8.4: February 23, 2023

- Message-ID header written to all outgoing email.
- Added diagnostics to decoding UTF-8 filenames.
- Fixed problem if SSL config string did not end with LF.
- Increased OAUTH2 password buffer to 2048 bytes.
- Added function seeMakeXOAuth2() that creates OAUTH2 password string.