SMTP/POP3/IMAP Email Engine Library for C/C++

Programmer's Manual

(SEE4C)

Version 8.3

March 21, 2022

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

MarshallSoft Computing, Inc.
Huntsville AL 35815

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

TABLE OF CONTENTS

1 Introduction

The **SMTP/POP3/IMAP Email Engine for C/C++** (**SEE4C**) library is a toolkit that allows software developers to quickly develop SMTP and POP3/IMAP email applications in C/C++, Visual C++, .NET and Visual C#.

The **SMTP/POP3/IMAP Email Engine** (SEE) is a component DLL 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.

A straight-forward 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.

This **SMTP/POP3/IMAP Programmers Manual** provides information need to compile and run programs in a C/C++ programming environment.

SEE4C includes more than 50 C/C++ example programs, including Visual Studio C++ and Visual Studio, which demonstrate SMTP and POP3/IMAP email functions.

The **SMTP/POP3/IMAP Email Engine for C/C++** component library supports and has been tested with C/C++, Microsoft Visual C++, Visual Studio .NET Framework (Visual C++ .NET, C# .NET), Borland C/C++,, Borland C++ Builder, Watcom C/C++, LCC-Win32 and MinGW C compilers. **SEE4C** can also be used with most other C/C++ Windows compilers.

SEE4C runs under all versions of Windows (Windows 95, Windows 98, Windows ME, Windows 2000, Windows 2003, Windows 2012, Windows NT, Windows XP, Vista, Windows7, Windows 8 and x64). The **SMTP/POP3/IMAP Email Engine SDK** DLLs (SEE32.DLL and SEE64.DLL) can also be used from any language (Visual Basic, Delphi, Visual FoxPro, COBOL, Xbase++, dBase, Microsoft Office, etc.) capable of calling the Windows API.

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

1. SEE4C is a standard Windows DLL (NOT an OCX or ActiveX control) and is much smaller than a comparable OCX or ActiveX control.
2. SEE4C does NOT depend on ActiveX or Microsoft Foundation Class (MFC) libraries or similar "support" libraries.
3. SEE is fully thread safe.
4. The SEE4C functions can be called from applications not capable of using controls.

MarshallSoft also has versions of the **SMTP/POP3/IMAP Email Engine** library for Visual Basic (SEE4VB), Delphi (SEE4D), PowerBASIC (SEE4PB),Visual FoxPro (SEE4FP), dBASE (SEE4DB), Cobol (SEE4CB) and Xbase++ (SEE4XB). 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 computer 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

Our goal is to provide a robust SMTP/POP3/IMAP email component library that you and your customers can depend upon. A fully functional evaluation version is available. Contact us if you have any questions.

1.1 Features

Some of the many features of the **SMTP/POP3 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.
- 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.
- 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.
- Decoding email from a File
- Specify SMTP or POP3 port.
- 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.
- Construct non-standard email that must be quoted (ex., EDIFACT)
- Includes over 52 functions for SMTP and POP3/IMAP control.
- Dozens of switches provided to control how email is sent or received.
- Supports setting priority via X-Priority header field.
- Removes contents of attachments before writing to disk.
- Use with servers requiring SSL with (free) STUNNEL proxy server.
- Can be used from GUI mode or console mode programs.
- Is fully thread safe.
- Both Win32 and Win64 DLLs included.
- Is native Windows code but can also be called from managed code.
- Will run on machines with or without .NET installed
- Implemented as a **standard** Windows DLL, which will work with all versions of Windows.
- Works with all versions of Microsoft Visual C++ (V4.0 through Visual Studio 2022)
- Supports most Windows C/C++ compilers (Borland, Watcom, MinGW C++, LCC-WIN32, etc.).
- Can be used with Microsoft Visual Studio .NET and Visual C# (Managed Code)
- Can be used with Microsoft Foundation Class (MFC), and Borland C++ Builder programs.
- 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 Basic, VB.NET, Visual FoxPro, Delphi, Xbase++, dBASE, COBOL, Access and Excel.
- Supports 32-bit and 64-bit Windows through Windows 11.
- License covers all programming languages.
- Royalty free distribution with your compiled application.
- Free technical support and downloadable updates for one year.
- Online documentation as well as in printable format.
- Evaluation versions are fully functional. No unlock code is required.

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

```
Legend: VS = Visual Studio, SSL = requires SSL connection, Con = Console Mode, GUI = GUI
Mode

[PROGRAM]      [DESCRIPTION]

Bcast       :  Sends bulk email to one recipient per email. (Con)
CodeTest    :  Base64 encodes/decodes strings. (Con)
CS_Mail     :  Sends email with optional attachment. (C#, console mode)
CS_SEE_VERS :  Displays SEE Version/Build #. (C#, GUI)
CS_Status   :  Similar to FROM and STATUS.  (C#, GUI)
Delete      :  Deletes email message from the server.  (Con)
Forward     :  Forwards undecoded email. (Con)
From        :  Displays header information for email on server.  (Con)
GB2312      :  Sends email that is GB2312 (simplified Chinese). (Con)
GetDOC      :  Automatic email document retrieval system. (Con)
GetRaw      :  Downloads specified email without decoding.  (Con)
Hello       :  Displays SEE Version/Build # using fsee class. (Con)
ImapFlagsT  :  Tests manipulation of flaps on IMAP server.  (Con)
ImapMBTest  :  Tests IMAP functions.  (Con)
ImapSearch  :  Tests IMAP search capability.  (Con)
ISO8859     :  Sends ISO-8859 encoded message and subject line (Con)
Mailer      :  Sends email with optional attachment. (Con)
MailHTML    :  Sends html encoded email with attachments. (GUI)
MailSSL     :  Connects to server required to send email. (SSL, Con)
MailTo      :  Sends email entered into dialog box.  (GUI)
MFC_PGM     :  Sends email. (Microsoft Foundation Class (MFC)
Mparts      :  Sends multipart MIME email. (Con)
POP3Rd      :  Specifies email message file to decode. (Con)
QM          :  Sends email. (Borland C++ Builder Win32 GUI)
Reader      :  Downloads email & attachments and saves to disk. (Con)
ReadHTML    :  Reads incoming HTML email & launches browser.  (Con)
Reads       :  Similar to READER but reads all email from server. (Con)
ReadSSL     :  Downloads email from POP3 server. (SSL, Con)
SEE_VERS    :  Displays SEE Version/Build #. (GUI)
SEEVER      :  Displays SEE Version/Build #. (Con)
SendHTML    :  Sends HTML email with embedded graphics attachments. (Con)
SendUTF8    :  Sends UTF-8 encoded email and subject line.  (Con)
SmtpOut     :  Writes email to file instead of sending. (Con)
Stat        :  Returns number of messages on POP3 server. (Con)
STATUS      :  Similar to FROM. (GUI)
Thread      :  Uses threads to check multiple mailboxes simultaneously. (Con)
VC_Mail     :  Sends email with optional attachment. (VS, Con)
VC_MailSSL  :  Sends email with optional attachment to send email. (VS, SSL, Con)
VC_Read     :  Downloads email & attachments and saves to disk. (VS, Con)
VC_ReadSSL  :  Downloads email & attachments and saves to disk. (VS, SSL, Con)
VC_Status   :  Returns number of messages on POP3 server. (VS, Con)
VC_StatusSSL:  Returns number of messages on POP3 server. (VS, SSL, Con)
VC_SEE_VERS :  Displays SEE Version/Build #. (VS, GUI)
```

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_4C) in the set.

- SEE4C Programmer's Manual (SEE_4C.PDF)
- SEE User's Manual (SEE_USR.PDF)
- SEE Reference Manual (SEE_REF.PDF)

The **SMTP/POP3/IMAP Programmer's Manual** (SEE_4C .PDF) is the language specific (C/C++) manual. All language dependent programming issues such as compiling, compilers and example programs are discussed in this manual.

The **SMTP/POP3/IMAP User's Manual** (SEE_USR.PDF) discusses SMTP and POP3/IMAP email processing as well as language independent programming issues such as application notes. Purchasing and licensing information is also provided.

The **SMTP/POP3/IMAP Reference Manual** (SEE_REF.PDF) contains details on each individual **SEE** function and provides a list of **SEE** error codes.

The online documentation can be accessed on the **SMTP/POP3/IMAP Email Engine for C/C++** product page at:

http://www.marshallsoft.com/see4c.htm

1.3 Example Program

The following example demonstrates the use of some of the **SMTP/POP3/IMAP Email** library functions:

```
#include <windows.h>
#include <stdio.h>
#include "see.h"
void main(int argc, char *argv  )
{int Code;
 // attach SEE library
 Code = seeAttach(1, 0);            // evaluation keycode is 0
 if(Code<0)
   {printf("Cannot attach SEE\n");
    exit(1);
   }
 // connect to SMTP mail server
 Code = seeSmtpConnect(
   0,                               // channel 0
   (char *)"mail.yourserver.net",   // your SMTP server
   (char *)"<user@domain.com>",     // your email address
   (char *)NULL);                   // Reply-To header
 if(Code<0)
   {printf("Connect failed\n");
    seeRelease();
    exit(1);
   }
 // send email
 Code = seeSendEmail(
   0,                                  // channel 0
   (char *)"<support@marshallsoft.com>", // To list
   (char *)NULL,                       // CC list
   (char *)NULL,                       // BCC list
   (char *)"SEE Test",                 // Subject
   (char *)"This is a test.",          // Message text
   (char *)"SEE4C75.ZIP");             // MIME attachment
 if(Code<0) printf("email NOT sent\n");
 // close connection to server
 seeClose(0);
 seeRelease();
}
```

In the example program above, **seeAttach** is called to initialize **SEE** and then **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. All fields in **seeSendEmail** are optional except the first.

After returning from **seeSendEmail**, the **seeClose function** is called to close the connection to the SMTP server. Lastly, **seeRelease** is called to perform **SEE** termination processing and release the Winsock.

See Section 1.1, "Features" for a list of all examples.

1.4 Installation

(1) Before installation of **SEE4C**, the Windows C/C++ compiler should already be installed on your system and tested. In particular, include command line tools when installing the compiler if you want to compile using command line makefiles. For help with makefiles, see MAKEFILE.TXT.

(2) Unzip SEE4C83.ZIP (evaluation version) or SEExxxx.ZIP (registered version, where xxxxx is your Customer ID) using any Windows unzip program..

(3) Run the installation program SETUP.EXE which will install all SEE4C files, including copying SEE32.DLL and SEE64.DLL to the Windows directory. Note that no DLL registration is required.

All recent WIN32 C/C++ compilers support the "declspec" keyword. Microsoft Visual C++ (version 4.0 and up), Borland (version 5.0 and 5.5), and LCC-Win32 compilers support the "declspec" keyword.

1.5 Uninstalling

Uninstalling SEE4C is very easy. First, run UNINSTAL.BAT (or UINSTALL.BAT), which will delete SEE32.DLL and SEE64.DLL from the Windows directory, typically C:\WINDOWS for Windows. Next delete the SEE4C project directory created when installing SEE4C.

1.6 Pricing

A developer license for the SMTP/POP3/IMAP Email Library can be registered for $119 USD. Purchasing details can be found in Section 1.4, "How to Purchase", of the SEE User's Manual (SEE_USR). (http:/ww.marshallsoft.com/see_usr.pdf)

Also see INVOICE.TXT or

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

1.7 Updates

When a developer license is purchased, the developer will be sent a set of registered DLLs plus a license file (SEExxxx.LIC). 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. The license can be updated for $33 if ordered within one year of the original purchase (or previous update). Between one year and three years, licenses can be updated for $55. After three years, updates are $77.

Note that the registered DLLs, (SEE32.DLL and SEE64.DLL), never expire.

2 Library Overview

The **SMTP/POP3/IMAP Email** component library has been tested on multiple computers running Windows XP through Windows 10..

The SEE4C library has also been tested with several C/C++ compilers, including Microsoft Visual C++ (all versions including Visual Studio C++ and Visual Studio C#), Borland C/C++, Borland C++ Builder, Turbo C/C++ for Windows, MinGW C++ and Watcom C/C++.

The SETUP installation program will copy the Lib's and DLL's to the Windows directory. Refer to Section 1.4 "Installation".

After SETUP is run, the SEE4C files are copied to the directory specified (default \SEE4C). Three sub-directories are created, as follows:

```
DOCS - All documentation files
APPS - All example code
DLLS - All DLL's
```

2.1 Keycode (License Key)

SEE32.DLL and SEE64.DLL each has a keycode encoded within it. The keycode is a 9 or 10 digit decimal number (unless it is 0), and will be found in the file KEYCODE.H. The keycode for the evaluation version is 0. You will receive a new keycode and a set of new DLL's when registering. 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 purchasing a license, it is best to remove the evaluation versions of the SEE64.DLL and SEE32.DLLs from the Windows search path or delete them before installing the purchased version

2.2 Win32 STDCALL and DECLSPEC

SEE32 is compiled using the _stdcall and _declspec keywords. This means that they use the same calling conventions and file naming conventions as the Win32 API. In particular, function names are NOT decorated. There are no leading underscores or trailing "@size" strings added to function names.

Microsoft Visual C/C++ users can look at the DLL function names using the dumpbin.exe executable:

```
dumpbin /exports SEE32.dll
```

2.3 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".

The following files can be found in the DLL sub-directory when SETUP is run:

```
see32.dll    - Win32 version of SEE
see64.dll    - Win64 version of SEE
```

The following files can be found in the APPS sub-directory when SETUP is run:

```
see32.lib    - LIB file corresponding to see32.dll
see64.lib    - LIB file corresponding to see64.dll
see32.obj    - Win32 object file (for static linking)
see64.obj    - Win64 object file (for static linking)
see32dm.obj  - Digital Mars Win32 object file (for static linking
```

2.4 Console Mode

SEE4C functions can be called from console mode programs. A "console mode" program is a Windows WIN32 or 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 WIN32/WIN64 API and the entire Windows address space. Programming using console mode programs reduces the complexity of using GUI code. All console mode programs can be converted to GUI mode by adding the necessary Windows GUI interface code

2.5 Using Threads

SEE4C is thread safe. Refer to the THREAD.C example program, which demonstrates the use of multiple threads. THREAD checks several POP3 mailboxes simultaneously.

2.6 Calling SEE functions from Visual C++

Like Windows itself, **SEE** functions are coded in ANSI C, but they can be called directly from both ANSI C programs and from C++ programs.

SEE functions can also be called using the C++ class wrapper **fsee**. See **fsee.cpp** and **fsee.h**. Refer to HELLO.CPP for an example.

2.7 Adding SEE Functions to an Existing Program

In order to call **SEE** functions from an existing program, (1) add

```
#include "see.h"
```

to your application source code, (2) link with SEE32.LIB (for MSVC), SEE32BC5.LIB (Borland C/C++ and C++ Builder), SEE32.LIB (Watcom), or SEE32LCC (Win32/LCC), and recompile from source.

For Win64, link with SEE64.LIB rather than SEE32.LIB

2.8 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.

2.9 Explicitly Loading a SEE DLL

When an application program runs that makes calls to SEE32.DLL (or SEE64.DLL), the Windows operating system will locate SEE32.DLL (or SEE64.DLL) by searching the directories as specified by the Windows search path. If the SEE32.DLL (or SEE64.DLL) is placed in the \WINDOWS directory (or \WINNT for Windows NT/2000), it will always be found by Windows.

However, SEE32.DLL (or SEE64.DLL) can also be loaded from a specified directory by using the GetProcAddress API function. For an example, refer to the LoadLib.c program.

2.10 Targeting a 64-Bit CPU

If a compiler generates 32-bit application code and is running on a 64-bit version of Windows, then compiling and linking is the same as it is on a 32-bit Windows system. The 32-bit application code generated will be executed by the Windows **WOW64** (Windows on Windows 64-bit) component.

If a compiler generates 64-bit application code and is running on a 64-bit version of Windows, then the compiler must be reconfigured to generate 32-bit application code if the application will call 32-bit DLL's such as SEE32.DLL. The 32-bit application code generated will be executed by the Windows **WOW64** (Windows on Windows 64-bit) component.

Visual Studio C/C++: Versions 2005 through 2015

With a project selected in Solution Explorer, on the Project menu, click Properties.
Click the "Configuration Manager" button in upper right corner.
Click the drop-down button below "Platform".
Click <New...>, then choose "x86" (Win32).

2.11 64-bit SEE

64-bit DLL's (such as SEE64.DLL) may only be used by 64-bit application programs running on 64-bit Windows computers. This means that 64-bit application programs must be linked with SEE64.LIB instead of SEE32.LIB.

However, if a compiler generates 32-bit code, the application must be linked with SEE32.DLL even though it may be running on a 64-bit machine.

There are numerous SEE4C 64-bit example programs. 64-bit Visual Studio 2008 through 2022 project files include the following in the /APPS folder:

```
vc_see_vers(VS2008)x64.vcproj
vc_mail(VS2008)x64.vcproj
vc_status(VS2008)x64.vcproj

through …

vc_see_vers(VS2022)x64.vcxproj
vc_mail(VS2022)x64.vcxproj
vc_status(VS2022)x64.vcxproj
```

2.12 32-bit Multitasking

SEE32.DLL is built to support multi-tasking.

3 Compiler Issues

Application programs can be compiled using an IDE or command line compiler tools. The following sections provide general compiler information.

3.1 Compiling Using an IDE.

All current windows compilers have an "Integrated Development Environment" (IDE) for building application programs in the Windows environment. Since there is not a standard format for IDE project files, file names must be entered into the IDE from the keyboard.

Note that IDES vary between the different compiler manufacturers and that IDES can also vary from version to version for the same compiler.

Creating a project makefile for the examples that have only command line makefiles is fairly straightforward. To begin, create a new project. For console mode applications, be sure to choose a console mode project type. For GUI applications, choose a GUI mode project type.

All of the IDE's use the concept of a file hierarchy. For example, the STAT example program file hierarchy in the IDE (for 32-bit) should look like:

```
STAT.EXE
  +++ STAT.C
  +++ SEE32.LIB
```

Replace SEE32.LIB with SEE32BC5.LIB if using Borland C++ Builder, and with SEE32LCC.LIB if using LCC-Win32.

The order of the files is not significant. Also refer to the sections on individual IDE's that follow this section.

3.1.1 Compiling Example Programs with an IDE

Most of the example programs can be compiled from your compiler's IDE. For Visual C++, "project makefiles" are used since they can be used by all versions of Visual C++ (v4.0, v5.0, v6.0, v.9.0, etc.). When opening the workspace, select "makefiles(.mak)" for the file type.

Alternatively, for VC++ v6.0, select "projects (.dsp)" for the file type. Visual Studio may also load ".dsp" project files as well as ".vcproj" project files.

3.2 Command Line Tool Setup

Many software developers overlook the power of using command line compilers. There are a number of very significant advantages to using the command line version of your C/C++ compiler. Among these are:

- **Easy of Use**: Once set up, typing a single key can compile one or a thousand programs.
- **Power**: Using the command line allows the use of batch files, automating complicated builds.
- **Compatibility**: Command line makefiles (unlike IDE project files) are normally compatible from one version of your compiler to the next.

If you want to compile from the command line, your command line compiler tools must be set up properly. Note that you have an option of installing the command line tools (or not) when your compiler is first installed. Refer to your compiler manufacturer's manual for details.

If necessary, increase the size of the environment table space (to 1024 for example) by adding

```
SHELL=C:\COMMAND.COM /e:1024 /p
```

to CONFIG.SYS in C:\ and then restart the computer. Yes, this works for all versions of Windows.

For all compilers, the path should point to the compiler's BIN directory. For example, to add "C:\BC50\BIN" to the existing path, use

```
PATH C:\BC50\BIN;%PATH%
```

Also note that console mode programs work the same on 64-bit machines as 32-bit machines.

3.2.1 Microsoft Visual C++

Set LIB and INCLUDE environment variables. For example,

```
SET INCLUDE=C:\MSVC\INCLUDE
SET LIB=C:\MSVC\LIB
```

3.2.2 Borland C++

Check that TURBOC.CFG, BCC32.CFG, TLINK.CFG, and TLINK32.CFG all have the correct information in them, as they should have when the compiler was installed. For example, assuming the BC compiler is installed at C:\BC5, the INCLUDE `(-I)` and LIB `(-L)` paths are specified by:

```
-IC:\BC5\INCLUDE
-LC:\BC5\LIB
```

BRCC (the Borland Resource Compiler) doesn't use the *.CFG files. Set the INCLUDE environment variable or BRCC will not be able to find the INCLUDE files (such as WINDOWS.H). For example,

```
SET INCLUDE=C:\BC5\INCLUDE
```

Clear the LIB environment variable (so it is not present when SET is typed at the command line) with

```
SET LIB=
```

3.2.3 Watcom C/C++

Set the WATCOM environment variables to point to the compilers include (H) and BIN directories. For example,

```
SET INCLUDE=C:\WC11\H;C:\WC11\H\NT
SET WATCOM=C:\WC11
SET EDPATH=C:\WC11\EDDAT
SET WWINHELP=E:\BINW
```

3.2.4 LCC-Win32 C

The LCC environment variables are set like the others. For example,

```
SET INCLUDE=C:\LCC\INCLUDE
SET LIB=C:\LCC\LIB
```

After making the above changes for the compiler, type PATH at the command line prompt to verify the search path, and type SET at the command line prompt to verify the INCLUDE and LIB environment variables.

3.3 Command Line Batch Files

If the compiler installation includes command line tools, then all of the example programs can be compiled directly from the command line. These same compiler commands can also be placed in a batch file.

See MAILER$.bat for an example of a console mode command line batch file and FROM$.bat for an example of a GUI mode command line batch file. Similarly, command line batch files can be created for all of the example programs.

3.4 Command Line Makefiles

Command line makefiles originated on UNIX systems. They are the standard way that C/C++ programs are constructed in command line environments. The advantage of makefiles (as compared to an integrated development environment) is that all compiler switches are coded within the makefile and the makefile can be run with a single keystroke.

Command line makefiles are provided for Microsoft, Borland, Watcom, , GCC, Digital Mars, and LCC command line compilers. They can be found in the APPS sub-directory:

- `Makefiles(Microsoft).zip` Microsoft C/C++ makefiles.
- `Makefiles(GCC).zip` MinGW GCC makefiles.

4.0 Supported Compilers

The **SMTP/POP3 Email Library for C/C++ (SEE4C)** has been tested and works with all versions of the following compilers:

- Microsoft Visual C++ (4.0 through 11.00)
- Visual C++ .Net through Visual Studio 2022)
- Visual C#,
- Borland C/C++
- Borland C++ Builder
- Embarcadero C/C++
- Watcom C/C++
- MinGW C++
- LCC C/C++
- Digital Mars C/C++

Other Windows C/C++ compilers may work as well. Refer also to Section 4.0, "Compiling Example Programs".

4.1 Microsoft Visual C/C++ (all versions)

Microsoft Visual C/C++ programs can be compiled from either the command line or from within the Microsoft development environment.

4.1.1 Microsoft Command Line Makefiles

Programs can be compiled using command line makefiles. All Microsoft command line makefiles end with "'_M_". To compile using a makefile, use the Microsoft NMAKE utility. For example,

```
NMAKE -f SEEVER(LIB)32._M_ : links see32.lib, which uses see32.dll
NMAKE -f SEEVER(OBJ)32._M_ : links see32.obj
NMAKE -f SEEVER(LIB)64._M_ : links see64.lib, which uses see64.dll
NMAKE -f SEEVER(OBJ)64._M_ : links see64.obj
```

The file `Makefiles(Microsoft).zip` contains the Microsoft Visual C/C++ command line makefiles.

Also note that console mode programs work the same on 64-bit machines as 32-bit machines.

4.1.2 Microsoft Developer Studio (VC v4.0)

To open an existing project, choose "File", then "Open Workspace", and then select "makefiles" from the list of file types. Most of the example programs have Microsoft Developer C/C++ project makefiles, ending with ".MAK" , such as SEEVER.MAK

To create a new project in MSVC v4.0, choose "File", then "New", then "Project Workspace". Select "Application" or "Console Application" for "Type:" and the project name for "Name". Choose Win32 for platform. Then select "Create". Select "Insert", then "Files into Project". Add all filenames including any resource file (.RC) and SEE32.LIB. Lastly, select "Build", then "Rebuild All". Be sure to specify /YX rather than /Yu in the project settings [Build, Settings…, C/C++].

4.1.3 Microsoft Developer Studio (VC v5.0)

To open an existing project, choose "File", then "Open Workspace", and then select "makefiles" from the list of file types. Most of the example programs have Microsoft Developer C/C++ project makefiles, ending with ".MAK" , such as SEEVER.MAK.

To create a new project in MSVC v5.0, choose "File", then "New", then "Win32 Application" or "Win32 Console Application " and the project name. Check "Create new workspace". Select "Project", then "Add to Project". Add all filenames including any resource file (.RC) and SEE32.LIB. Lastly, select "Rebuild All".

If the compiler complains that it cannot find "_main", "Console Application" was chosen but the program being compiled is a GUI application. If the compiler complains that it cannot find "WinMain", "Application" was chosen but the program being compiled is a console mode application. Be sure to specify /YX rather than /Yu in your project settings [Build, Settings…, C/C++].

4.1.4 Microsoft Visual Studio (VC v6.0)

To open an existing project, follow the same directions as for MSVC v5.0, except that a DSP project file may be used instead of the MAK project makefile.

To create a new project in MSVC v6.0, follow the same directions as for MSVC v5.0 above.

4.1.5 Microsoft Visual Studio 2003 and 2005

Open the VC project file (files ending in ".vcproj" or ".dsp").

4.1.6 Microsoft Visual Studio 2008

Visual Studio 2008 specific project files end with ".(VS2008).vcproj" for 32-bit applications and ".(VS2008)x64.vcproj" for 64-bit applications.

4.1.7 Microsoft Visual Studio 2010

Visual Studio 2010 specific project files end with ".(VS2010).vcxproj" for 32-bit applications and ".(VS2010)x64.vcxproj" for 64-bit applications.

Note that project files for Visual Studio 2010 end with ".vcxproj" rather than ".vcproj" as in earlier versions of Visual Studio.

4.1.8 Microsoft Visual Studio 2012 through 2022

Visual Studio 2012 specific project files end with ".(VS2012).vcxproj" for 32-bit applications and ".(VS2012)x64.vcxproj" for 64-bit applications. Visual Studio 2013 specific project files end with ".(VS2013).vcxproj" for 32-bit applications and ".(VS2013)x64.vcxproj" for 64-bit applications.

In order to convert an older project file to VS 2012 through VS 2022:

1. Open the older (.dsp, .vcproj, .vcxproj) project file with VS 2012 or VS 2013.
2. Let VS 2012 (2013) update to 2012 (2013) format when prompted.
3. Select "Project", "Properties", "Linker", then "Advanced".
4. Change the "Image Has Safe Exception Handlers" to NO (/SAFESEH:NO)
5. Save project. This will insert the line

```
<ImageHasSafeExceptionHandlers>false</ImageHasSafeExceptionHandlers>
```

into your project file as the last line before `</Link>`

4.1.9 Microsoft C++ Express Edition

The "Express Edition" of Microsoft Visual Studio is available as a free download at
http://www.microsoft.com/express/download/

Open the VC project file (files ending in ".vcxproj", ".vcproj" or ".dsp"), build and run, as in previous versions of Visual Studio. Also see section 4.2 below.

4.2 Microsoft Visual Studio C++ .NET

All Microsoft Visual C++ .NET and Visual Studio C++ projects end with the extension ".vcproj". For example,

```
vc_vers.vcproj
```

In order to open an existing Visual C++ project, choose "File", "Open", and then "Project" from the Visual Studio menu. Specify the directory containing the Visual C++ project files (for example, C:\SEE4C\APPS).

In order to call **SEE** functions from Visual Studio C++ programs, do the following to the existing Visual Studio C++ project:

(1) Add

```
#include "see.h"
#include "keycode.h"
```

after the existing #include statements.

(2) Open the project properties window under "Project" on the main menu. If the project is named "MyProject", then select "MyProject properties".

(3) Select "Configuration Properties", "Linker", "Input", "Additional Dependencies", and then type in "SEE32.LIB" into the edit box. Note that SEE32.LIB must be in the project directory along with all of the source files.

(4) Rebuild.

NOTE: If using pre-compiled headers, the include statement

```
#include "stdafx.h"
```

must be the first include statement in the program.

4.3 Microsoft Visual Studio C#

SMTP/POP3/IMAP Email functions can be called from Microsoft Visual C# (C-sharp) in the same manner as Win32 API functions.

All Visual Studio C# projects end with extension ".csproj". For example,

```
cs_vers.csproj
```

In order to open an existing C# project, choose "File", "Open", and then "Project" from the Microsoft C# Development Environment. Specify the directory containing the C# project files (for example, C:\SEE4C\APPS).

In order to call **SEE** functions from an existing Visual C# programs, do the following to the C# source code:

(1) Add the contents of file `see_funs.cs` to the source code after

```
public class see : System.Windows.Forms.Form
```

(2) Add the constants from `see_cons.cs` to the program as they are needed.

Look at the `cs_vers` program in the \APPS sub-directory for an example.

(3) Set "unsafe" compilation. Verify that AllowUnsafeBlocks is set to true in the project file (.vcproj).

See the `CS_MAIL` example project.

4.4 Borland C/C++

Email us at info@marshallsoft.com with subject

MSC HELP Borland

for example Borland 5.5 programs

.

4.5 Borland C++ Builder

Email us at info@marshallsoft.com with subject

MSC HELP Borland Builder

for example Borland C++ Builder programs.

4.6 Watcom C/C++

Email us at info@marshallsoft.com with subject

MSC HELP Watcom 11

for example Watcom 11 programs.

4.7 LCC-Win32 C

Email us at info@marshallsoft.com with subject

MSC HELP LCC

for example LCC32 programs.

4.8 MinGW C/C++

MinGW (Minimalist GNU for Windows) is part of the GNU Compiler Collection (GCC), and GNU Binutils, for use in the development of native Microsoft Windows applications. See http://www.mingw.org

Console mode programs are compiled from the command line; for example

```
gcc -Wall server.c csc32.lib -o server.exe
```

The current version of MinGW does not come with a resource compiler (for RC files), so GUI Window applications that use resource files cannot be compiled with MinGW.

All GCC command line makefiles end with "_G", as for example

```
make -f fcever(lib)32._G_
```

Command line makefiles for GCC can be found in the archive `Makefiles(GCC).zip`

4.9 Digital Mars

Digital Mars C/C++ (see http://www.digitalmars.com) programs can be compiled from the command line by using the Digital Mars "make" program.

All Digital Mars command line makefiles end with "_D_". For example,

```
make -f fcever(obj)32._D_
```

Command line makefiles for Digital Mars can be found in the archive `Makefiles(Mars).zip`

4.10 Embarcadero C/C++

http://docwiki.embarcadero.com/RADStudio/Sydney/en/MKEXP.EXE,_the_64-bit_Windows_Import_Library_Tool_for_C++

MKEXP.EXE, the 64-bit Windows Import Library Tool for C++
Go Up to Command-Line Utilities Index
http://docwiki.embarcadero.com/RADStudio/Sydney/en/Command-Line_Utilities_Index

MKEXP.EXE is the 64-bit Windows counterpart of IMPLIB.EXE, the Import Library Tool for Win32.

http://docwiki.embarcadero.com/RADStudio/Sydney/en/IMPLIB.EXE,_the_Import_Library_Tool_for_Win32

However, MKEXP does not support all the same options supported by IMPLIB.

MKEXP produces GNU-style (ELF format) archive files (.a files).

You can use .DLL, .DEF, or OMF files with MKEXP.

Examples

Generating an import library from a DLL:

```
mkexp see64.a see64.dll
```

Generating an import library from a .DEF file:

```
mkexp see64.a see64.def
```

The resulting .a files are included in the library section of the link command line for any application or DLL that uses see64.dll. The .a files that are produced by mkexp are GNU style archive files containing a single object file with an Embarcadero specific set of data to describe all the exports from the DLL.

5 Compiling Example Programs

Most of the example programs are written in console mode. This was done in order to provide the clearest possible code, without the complication and complexity of GUI code. All console mode programs can be converted to GUI mode by coding the necessary windows code

5.1 Static Libraries

The developer can also statically link SEE32.OBJ (or SEE64.OBJ) rather than making calls to SEE32.DLL. Makefiles that statically link contain "(OBJ)" in the makefile name, such as

```
seever(obj)32._D_
seever(obj)32._M_
seever(obj)64._M_
```

To create an application that links SEE32.OBJ statically:

(1) All application code that includes SEE.H must define STATIC_LIBRARY before including SEE.H
(2) The application must link with WSOCK32.

If using Microsoft Developer Studio, make these changes:

(1) To the project file: Do NOT add SEE32.LIB to your project file.
(2) To the settings: (See "Build Settings" or "Project/Settings")
(2a) C/C++ Tab: Add STATIC_LIBRARY to "preprocessor definitions:".
(2b) Link Tab: Add wsock32.lib and see32.obj to "object/library modules:".
(3) Add #include "see.h" to all source files that make calls to SEE functions.

Static libraries for 64-bit applications are built in the same way as for 32-bit applications, except that one links with SEE64.OBJ rather than SEE32.OBJ..

6 Example Programs.

The example programs can be compiled by using either the command line compiler or the compiler integrated development environment (IDE). Most compiler vendors provide both IDE and command line tools, although some compilers are command line only (Borland C/C++ 5.5 and LCC-Win32) or IDE only (Borland C/C++ Builder).

Some of the example programs are written in GUI mode, although most are written in console mode. Note that console mode programs must be run from the Windows command prompt. Also note that console mode programs can be converted to GUI mode by adding the necessary Windows interface code.

Refer also to Section 4, "Supported Compilers".

6.1 Connectionless Example Programs

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

6.1.1 SEE_VERS

SEE_VERS is a Win32/Win64 GUI example program that displays the SEE Version and Build #.
Run this program to verify that SEE has been installed properly.

```
         GCC : SEE_Vers(lib)32._G_
   Microsoft : SEE_Vers(lib)32._M_
   Microsoft : SEE_Vers(lib)64._M_

Digital Mars : SEE_Vers(obj)32._D_
   Microsoft : SEE_Vers(obj)32._M_
   Microsoft : SEE_Vers(obj)64._M_

     Open workspace see_vers.mak in VC IDE.
     Open workspace see_vers.dsp in VC (ver 6.0 only).
```

6.1.2 VC_SEE_VERS

The VC_SEE_VERS example GUI mode program is the Visual Studio C++ version of the SEE_VERS example.

Open project VC_SEE_VERS.VCPROJ in Visual Studio C++.

6.1.3 CS_SEE_VERS

The CS_SEE_VERS example GUI mode program is the C# (C-sharp) version of the SEE_VERS example.

Open project CS_SEE_VERS.CSPROJ in Visual Studio C#.

6.1.4 SEEVER

The SEEVER example is the console mode version of the SEE_VERS example. It displays the **SEE** library version number and registration string. Its purpose is to display the **SEE** version, build, and registration string as well as to verify that SEE32.DLL/SEE64.DLL is being found and loaded by Windows.

SEEVER can be compiled from either the command line or from Microsoft Visual Studio (SEEVER32.MAK).

6.1.5 Hello

The HELLO example console mode program is a **C++ version** of SEEVER that demonstrates the use of the **fsee** class.

6.1.6 CodeTest

The CODETEST example console mode program demonstrates how to use **seeEncodeBuffer** and **seeDecodeBuffer**, which BASE64 encodes and decodes several test strings. It also has an example of encoding and decoding UTF-8 strings.

6.1.7 Pop3Rd

The Pop3Rd example console mode program uses the **seePop3Source** function to specify an (undecoded) email message file to be decoded.

6.1.8 TestConn

The TestConn example console mode 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.

6.2 SMTP Programs

There are 8 SMTP example programs. SMTP programs send email.

6.2.1 Forward

The FORWARD example console mode program forwards an (undecoded) email to a new recipient.

6.2.2 GB2312

The GB2312 example console mode 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).

6.2.3 MailSSL

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

6.2.4 ISO8859

The ISO8859 console mode program sends a text message and a 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).

6.2.5 Mailer

MAILER is a console mode program that emails a message, taking its input from the command line. For example, to email the file TEST.TXT with subject "test" to support@marshallsoft.com with (optional) attachment TEST.ZIP, type

```
MAILER @test.txt Test "<info@marshallsoft.com>" test.zip
```

Note that <> brackets are required around the email address.

6.2.6 VC_Mail

The VC_MAIL example console mode program is the **Visual Studio C++** version of the MAILER example program in section 6.2.10 above.

Load the VC_MAIL.VCPROJ project in Visual Studio C++.

6.2.7 CS_Mail

The CS_MAIL example **GUI mode** program is the **Visual Studio C# (C-sharp)** version of the MAILER example program in section 6.2.10 above.

Load the project CS_MAIL.CSPROJ in C#.

6.2.8 MailHTML

The MailHTML example **GUI mode** program emails an HTML encoded message, with optional embedded images, alternate text, and additional attachments. All parameters are taken from a dialog box. Compare to the SendHTML example program.

6.2.9 MailTo

MailTo is a **Win32 GUI** (Graphical User Interface) application that emails a message entered into a dialog box. All necessary information is entered at runtime into dialog boxes. MailTo can be compiled from either the command line or from Microsoft Visual Studio (MailTo.MAK).

6.2.10 Mparts

The MParts console mode example program sends a multipart MIME email in which the Content-Type headers for each attachment are specified.

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

6.2.11 SendHTML

The SendHTML console mode example program sends an HTML email (with embedded graphics attachments), which can be displayed by any email client that is capable of rendering HTML, such as Outlook and Eudora. SendHTML takes no arguments. Compare to the MailHTML example program.

6.2.12 SendUTF8

The SendUTF8 console mode example program sends a text message and subject line that is UTF-8 encoded. The recipient's email client will be able to display the email message using the Unicode character set provided that it is capable of identifying UTF-8 MIME parts and is Unicode capable.

6.2.13 SmtpOut

The SMTPOut console mode example program constructs a specified email message, then writes it to a file rather than sending it to an SMTP server.

6.3 POP3/IMAP Example Programs

There are 9 POP3/IMAP example programs that read email.

6.3.1 ReadSSL

The ReadSSL example console mode program downloads email messages from a POP3 server that requires SSL, such as Gmail, Hotmail, Yahoo, and Microsoft Online.

6.3.2 Reader

READER is a console mode program that reads an email message, including any MIME attachments, saving it to disk. For example, to read email message #1 and save it as file MYMAIL.TXT,

```
READER 1 MYMAIL.TXT
```

Any attachments are saved to the current directory.

6.3.3 VC_Read

VC_Read is the Visual Studio version of the READER example program.

6.3.4 ReadHTML

The READHTML console mode example program reads an incoming HTML email message and then launches the internet browser in order to display the email as rendered HTML.

6.3.5 Reads

The READS console mode program is similar to READER except that it reads all email from the POP3 server. READS takes no arguments. Attachments (if any) are saved to the current directory.

6.3.6 Status

STATUS is the console mode equivalent of FROM. STATUS takes no arguments.

6.3.7 VC_Status

VC_Status is the Visual Studio version of the STATUS example program.

6.3.8 CS_Status

CS_Status is the C# version of the STATUS example program.

Load the project CS_STATUS.CSPROJ in C#.

6.3.9 Thread

THREAD is a console mode program that uses threads to check several POP3 mailboxes simultaneously. You must edit THREAD.C with the POP3 mailbox settings to check before compiling.

6.4 IMAP-Only Example Programs

There are three IMAP-only example programs. All of the POP3 programs in the above section will also work with IMAP servers after defining the symbol

```
CONNECT_TO_IMAP_SERVER
```

in the program source code before compiling.

6.4.1 ImapFlagsT

The ImapFlagsT console-mode 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.

6.4.2 ImapSearch

The ImapSearch console-mode example program tests IMAP search capability.

See ImapSearch.txt for a list of all IMAP search strings.

6.4.3 ImapMBtest

The ImapMBtest console-mode example program tests IMAP functions seeImapConnect, seeImapListMB, seeImapDeleteMB, seeImapCreateMB, and seeImapSelectMB.

7 Revision History

Version 1.0: June 1, 1998.
- The official release of version 1.0.

Version 2.0: September 14, 1998.
- A major update adding POP3 capability.
- Another SMTP example program (BCAST).

Version 2.1: November 16, 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.
- Added SEE_GET_ATTACH_COUNT and SEE_GET_LAST_RESPONSE.
- Added GETDOC, SEEVER, and VERUSR example programs.
- SMTP performance improved.
- Added seeEncodeBuffer and seeDecodeBuffer functions.

Version 3.0: April 1, 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.
- Win16 version can get time zone from TZ environment variable.
- Quoted-printable messages can handle soft line breaks.
- Quoted-printable message can handle embedded HTML text.

Version 3.1: July 12, 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.
- Don't write "+OK" line to email message file.
- Added seeExtractLine function.

Version 3.2: January 10, 2000.
- Added QUOTED_8859 processing (QUOTED_8859).
- 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 CODETEST examples.
- Added support for LCC-Win32.
- 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 12, 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 ISO, MAILS, SENDHTML, and READHTML example programs.
- Added SEE_IGNORE_REJECTED to ignore error returned if recipient is rejected.

Version 3.6: March 14, 2003

- Added seeSendHTML().
- 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 17, 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.
- Added ISO8859, GB2312, and MParts example programs.

Version 4.0: June 12, 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 Pop3Rd example programs added.
- Added function seeByteToShort
- Added function seeShortToByte

Version 5.0: April 8, 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: April 27, 2009 (Win32 and Win64 (x64)) 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) [bld 6]
- 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 to support x64.

Version 5.2: February 19, 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 1, 2010

- 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.
- Added support for MinGW GCC compiler.

Version 7.0: October 24, 2011

- 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
- NAME_BUF increased from 1024 to 5120 characters.
- 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()
- Use large buffer (64K) for IMAP server response on channel 0

Version 7.1: April 2, 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: August 23, 2013

- 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
- Added new example programs vc_Read and vc_ReadSSL.

Version 7.3: November 21, 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 strings.
- Added UTF8 quoting : seeIntegerParam(Chan, SEE_QUOTED_PRINTABLE, QUOTED_UTF8)

Version 7.4: March 21, 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 TestConn.c that tests connection to server.

Version 7.5: September 12, 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)
- Modify: The max connection attempts reduced from 5 to 3.
- Modify: The trace filename format changed to "%s_%1d.log" (eg: c:\see4c\ssl\rcpt_trace_0.log)
- 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 3, 2018

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

Version 8.1: February 21, 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 11, 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: March 21, 2022

- Protocol XOAUTH2 must be explicitly enabled - seeIntegerParam(0, SEE_AUTHENTICATE_PROTOCOL, AUTHENTICATE_XOAUTH2).
- 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.