Contents

Licensing & Disclaimer

Disclaimer

Evaluation Agreement

Ombudsman statement

Registration

Licensing

Overview

Basic SMTP Functions

Advance SMTP Functions

POP3 Functions

pop3_ReadAttached
pop3_ReadBodyText
pop3_ResultsSize
pop3_ResetMail
pop3_RetrieveMail
pop3_RetrieveIntoFile
pop3_SaveAttachedTo
pop3_SetProgressBar
pop3_SetTimeOut

Appendix

Disclaimer

H&S Technology, Inc. provides the software and documentation without warranty of any kind, either express or implied, including, but not limited to, any implied warranties of merchantability or fitness for a particular purpose or warranties of quality or performance. Your exclusive remedy with respect to the software, the documentation and this agreement will be limited to, at H&S Technology's option, either (a) repair or replace any defective Software or Documentation or (b) refund the original purchase price paid.

In no event will H&S Technology, Inc. and its employees be liable for any damages arising out of the use or inability to use the software or documentation, including, but not limited to, any damages for lost profits, anticipated benefits, or business interruption, even if they have been advised of the possibility of such damages.

Evaluation Agreement

H&S Technology, Inc. hereby grants you a 30 days limited license for purpose of evaluating the Software. After 30 days, you are required to register the product or completely remove the software from your machines.

Use of this software indicates your acceptance of these terms and conditions. If you do not agree with them, do not use the software.

Ombudsman statement

H&S Technology, Inc. is a member of the Association of Shareware Professionals (ASP). ASP wants to make sure that the shareware principle works for you. If you are unable to resolve a shareware-related problem with an ASP member by contacting the member directly, ASP may be able to help. The ASP Ombudsman can help you resolve a dispute or problem with an ASP member, but does not provide technical support for members' products. Please write to the ASP Ombudsman at 157-F Love Ave, Greenwood, IN 46142 USA, FAX 317-888-2195, or send email to omb@asp-shareware.org.

Registration

You will need to register to use the product after the 30 days trial period. If you registered, you will get a manual and the latest copy of the product. For registration fees and how to register, please refer to order.wri file for instructions and the latest prices.

Licensing

YOU SHOULD READ THE FOLLOWING TERMS AND CONDITIONS CAREFULLY BEFORE USING THIS SOFTWARE. USE OF THIS SOFTWARE INDICATES YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS. IF YOU DO NOT AGREE WITH THEM, DO NOT USE THE SOFTWARE.

THIS LICENSE AGREEMENT IS A LEGAL AGREEMENT BETWEEN YOU (EITHER AN INDIVIDUAL OR A LEGAL ENTITY) AND H&S TECHNOLOGY, INC.

DEFINITION(S)

1. "SOFTWARE" MEANS THE SMTPOP16 DLL OR SMTPOP32 DLL SOFTWARE PRODUCTS.

Developer license

A single developer license entitles only the registered user to use the SOFTWARE on a single machine. The SOFTWARE cannot be installed on network shared drives.

Runtime license

A runtime license entitles only the registered user to use the runtime DLL as a component of the derivative product of the SOFTWARE on a single machine. Runtime licenses can only be purchased for applications/products that were developed using the Developer's license.

Developer Server license

A developer server license entitles a registered copy on a file server for unlimited users whose workstations are directly connected to the server to access and use the SOFTWARE. The file server where the software is installed must be the registered users' primary file server.

Runtime Server license

A runtime server license extends the runtime license to a file server for unlimited users whose workstations are directly connected to the server to access. The file server where the application is installed must be the registered users' primary file server.

Developer Corporate license

A corporate license entitles a registered company to unlimited use of the SOFTWARE within the company.

Runtime Corporate license

A runtime corporate license extends the runtime license to its employees within the company.

Distribution license

A distribution license entitles a registered user to distribute unlimited copies of the runtime DLL with an

application derived from the software. The registered copy of the SOFTWARE is therefore licensed for use in conjunction with that application only.

Distributing DLL

You may distribute only the runtime DLL (SMTPOP16.DLL or SMTPOP32.DLL depending on the license you have purchased) with your application, provided: 1) the application is developed using a registered copy of the SOFTWARE; and 2) you have purchased the appropriate runtime licenses for the end users. However, you are not allowed to distribute any of the documentation or supporting files, e.g. *.hlp, *.h, *.bas, *.lib files.

You may not reverse engineer or disassemble the software, or sell the software. You may not distribute your SOFTWARE registration key and serial number under any circumstances.

For more information regarding licensing, please write to the address below:

H&S Technology, Inc.

P.O. Box 5152

Katy, TX 77491-5152

USA

Tel: (281) 395-0345

Fax: (281) 395-8884

Email: hnstech@compuserve.com

URL: http://www.hnstech.com

About

SMTPOP16.DLL/SMTPOP32.DLL version 2.10.

H&S Technology, Inc.

P.O. Box 5152

Katy,

TX 77491-5152

USA

Tel: (281) 395-0345

Fax: (281) 395-8884

E-mail: hnstech@compuserve.com.

URL: http://www.hnstech.com

Introduction

Welcome to H&S Technology, Inc. Simple Mail Transfer Protocol (SMTP) and Post Office Protocol version 3 (POP3) Application Programming Interface (API) for Windows development toolkit. This API allows you to integrate SMTP and POP3 functions into your application seamlessly. It is also MIME v1.0 compliance, thereby allowing you to send and receive MIME v1.0 compliance mails. All you need is a WINSOCK.DLL from your TCPIP software supplier.

The API comes in two versions;

- SMTPOP16.DLL for Windows 3.1 and Windows for Workgroups,

- SMTPOP32.DLL for Windows 95 and Windows NT.

Unless otherwise stated, both versions have the same functions and calling parameters.

Requirements

SMTPOP16.DLL requires WINSOCK.DLL in your search path and Windows 3.1 or Windows for Workgroups.

SMTPOP32.DLL requires WSOCK32.DLL in your search path and Windows 95 or Windows NT.

Installation

In order to use SMTPOP16.DLL or SMTPOP32.DLL, you will need to run setup to properly install the software. The DLL will not function properly unless you perform the following installation process correctly. Begin the installation from Windows.

Note: Please take a moment to locate your serial number and registration key on the installation diskette. If you are installing for evaluation only, then you do not need the serial number and registration key, just click OK when you are asked for serial number and registration key.

Related Topics:

Installing SMTPOP16.DLL.

Installing SMTPOP32.DLL.

Installing SMTPOP16.DLL.

To begin the install process from a distribution diskette:

1. Insert the SMTPOP16 disk in your floppy drive.

2. Select Run from the Program Manager File Menu.

3. Type X:\SETUP in the Command Line text box, substituting your floppy drive's letter for X (for example, if your floppy driver letter is A, type A:\SETUP).

4. Click OK.

5. Follow the instructions on the screens.

6. After you have completed the installation process, add SMTPOP16.DLL in your search PATH or copy it to your Windows directory.

To begin the install process from a SMTPOP16.ZIP file:

1. Create a temporary directory and copy the zip file into the

   temporary directory.

2. Unzip the file with PKUNZIP SMTPOP16.ZIP.

3. Select Run from the Program Manager File Menu.

4. Type <PATH>\SETUP in the Command Line text box, substituting the temporary directory for <PATH> (for example, if your temporary directory is C:\TEMP, type C:\TEMP\SETUP).

5. Click OK.

6. Follow the instructions on the screens.

7. After you have completed the installation process, add SMTPOP16.DLL in your search PATH or copy it to your Windows directory.

Installing SMTPOP32.DLL.

To begin the install process from a distribution diskette:

1. Insert the SMTPOP32 disk in your floppy drive.

2. If you are using Windows 95, select Start from the Taskbar, then

   Select Run from the Start menu.

   or

   If you are using Windows NT, select Run from the Program Manager File menu.

3. Type X:\SETUP, substituting your floppy drive's letter for X

   (for example, if your floppy driver letter is A, type A:\SETUP).

4. Click OK.

5. Follow the instructions on the screens.

6. After you have completed the installation process, add SMTPOP32.DLL in your search PATH or copy it to your Windows directory.

To begin the install process from a SMTPOP32.ZIP file:

1. Create a temporary directory and copy SMTPOP32.ZIP into the temporary directory.

2. Unzip the file with PKUNZIP SMTPOP32.ZIP.

3. If you are using Windows 95, select Start from the Taskbar, then

   Select Run from the Start menu.

   or

   If you are using Windows NT, select Run from the Program Manager File menu.

4. Type <PATH>\SETUP32 in the Command Line text box, substituting the temporary directory for <PATH> (for example, if your temporary directory is C:\TEMP, type C:\TEMP\SETUP).

5. Click OK.

6. Follow the instructions on the screens.

7. After you have completed the installation process, add SMTPOP32.DLL in your search PATH or copy it to your Windows directory.

Upgrades and Fixes

For upgrades and fixes, please download the software from CompuServe's WINSHARE forum or our website at http://www.hnstech.com and re-install the software. You will need your serial number and registration key to re-install the software, so please make sure you keep the serial number and registration key in a safe place.

Technical Support

For technical support and consulting, please write to:

Mail address:

H&S Technology, Inc.

P.O. Box 5152

Katy,

TX 77491-5152

USA

Tel: (281) 395-0345

Fax: (281) 395-8884

E-mail: hnstech@compuserve.com.

URL: http://www.hnstech.com

H&S Technology, Inc. provides free technical support to all registered users (excluding runtime users) for 90 days after registration. Please contact us for pricing on extended support. When requesting for service, please include the following information:

1) Your serial number.

2) Product and version.

3) O/S (Win 3.1, Win95, or NT),

4) Memory/RAM size,

5) TCP/IP stack and vendor names,

6) Winsock version, if known,

7) SMTPOP error number, if any,

8) Trace log, if any,

9) Development tool (VB 4.0, MSVC 4.0, PB 5.0, BC++, Delphi,....)

10) Your script/code if possible.

Comments and Suggestions

H&S Technology, Inc. welcomes any comments and suggestions you may have. We value your input to help us improve our products. Please feel free to send your comments and suggestions to 70531.1066@compuserve.com. Thank you.

Trademarks

Introduction

This section describes the basic SMTP functions. For each function, it gives a brief description of the function, the function prototype, and a simple example on how to use it. The function prototype describes, in C syntax, the calling sequence and parameters needed for the function to operate properly.

Please refer to the appendix for information on how to obtain function prototypes for other programming languages.

smtp_Abort

Description:

This function aborts the current mail transaction. All unsent mail information (i.e. recipients, mail data...) will be discarded. This will result in incomplete mail being sent.

Function Prototype :

```
long smtp_Abort(long lHandle)
```

Parameters:

lHandle	smtp session handle.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long hndl;
char buffer[512];

hndl = smtp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   // processing...
   smtp_Abort(hndl);
   // processing...
   smtp_Quit(hndl);
}
```

smtp_Expand

Description:

This function expands the mailing list. If successful, the full user name and email address will be returned and are separated by "\r\n". The return buffer size must be large enough to hold the results. A positive return value indicates the number of bytes truncated.

Function Prototype :

```
long smtp_Expand(long lHandle, char *maillist, char *buffer, long blen)
```

Parameters:

lHandle	smtp session handle.
maillist	a mailing list.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small.

Example:

```
long hndl;
char buffer[512];

hndl = smtp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   if ( smtp_Expand(hndl, "mymailist", buffer, 512) >= 0 )
   {
      // processing...
   }
   smtp_Quit(hndl);
}
```

smtp_GetReply

Description:

This function retrieves the current mail server's replies from the reply buffer. The return buffer size must be large enough to hold the results.

Function Prototype :

```
long smtp_GetReply(long lHandle, char *buffer, long blen)
```

Parameters:

lHandle	smtp session handle.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl;
char buffer[512];

hndl = smtp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   smtp_GetReply(hndl, buffer, 512);
   // other processing...
   smtp_Quit(hndl);
}
```

smtp_Help

Description:

This function requests help information from the SMTP mail server. If successful, the specified help command will be returned. The return buffer size must be large enough to hold the results.

Function Prototype :

```
long smtp_Help(long lHandle, char *helpcmd, char *buffer, long blen)
```

Parameters:

lHandle	smtp session handle.
helpcmd	help on command.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl;
char buffer[512];

hndl = smtp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   if ( smtp_Help(hndl, "", buffer, 512) >= 0 )
   {
      // processing...
   }
   smtp_Quit(hndl);
}
```

smtp_LastMsg

Description:

This function retrieves the last function message into a user-specified buffer. The message can be of information, warning, or error. It may not necessary be the message from the server. Please refer to Error Messages for detail of the messages. The return buffer string is null terminated.

Function Prototype :

```
long smtp_LastMsg(char *buffer, long blen)
```

Parameters:

buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl;
char msg[512];

hndl = smtp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   if ( smtp_SendMail(hndl, "to_users", "cc_users", "bcc_users",
        "subject", "hello, world", "", 0) == 0 )
   {
      // processing...
   }
   smtp_Quit(hndl);
}
else
   smtp_LastMsg(msg, 512);
```

smtp_Noop

Description:

This function sends a "no operation" command to the SMTP mail server. This function does not affect any parameters or previously called functions. The SMTP mail server does not perform any action other than acknowledging the call.

Function Prototype :

```
long smtp_Noop(long lHandle)
```

Parameters:

lHandle	smtp session handle.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long hndl;

hndl = smtp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   smtp_Noop(hndl);
   // processing...
   smtp_Quit(hndl);
}
```

smtp_LoginMail

Description:

This function establishes an SMTP session with the SMTP server (host) on the well known SMTP port. Mailserver name can be either IP address or Hostname. Username is optional, i.e. it can be an empty string.

Function Prototype :

```
long smtp_LoginMail(char *mailserver, char *username, char *mailid, char *domain)
```

Parameters:

mailserver	internet host name or IP address of your mail/smtp server
username	Your name. (optional, can be empty string).
mailid	Your smtp mail id.
domain	Your domain name (e.g. company.com).

Return Value:

Long. Returns a handle to the SMTP session if it succeeds and -1 if it fails. The valid handle is always greater than 0.

Example:

```
long hndl;

hndl = smtp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   if ( smtp_SendMail(hndl, "to_users", "cc_users", "bcc_users",
        "subject", "hello, world", "", 0) == 0 )
   {
      // processing...
   }
   smtp_Quit(hndl);
}
```

smtp_LoginMailPort

Description:

This function establishes an SMTP session with the SMTP server (host) on a given SMTP port. Mailserver name can be either IP address or Hostname. Username is optional, i.e. it can be an empty string.

Function Prototype :

```
long smtp_LoginMailPort(char *mailserver, long port, char *username,
char *mailid, char *domain)
```

Parameters:

mailserver	internet host name or IP address of your mail/smtp server
Port	The SMTP mail server port.
username	Your name. (optional, can be empty string).
mailid	Your smtp mail id.
domain	Your domain name (e.g. company.com).

Return Value:

Long. Returns a handle to the SMTP session if it succeeds and -1 if it fails. The valid handle is always greater than 0.

Example:

```
long hndl;

hndl = smtp_LoginMailPort("smtp_hostname", 1234, "myname", "myid",
"mydomain");
if ( hndl > 0 )
{
   if ( smtp_SendMail(hndl, "to_users", "cc_users", "bcc_users",
        "subject", "hello, world", "", 0) == 0 )
   {
      // processing...
   }
   smtp_Quit(hndl);
}
```

smtp_Quit

Description:

This function quits and disconnects from SMTP mail server, if connected, and releases the mail session control block.

Function Prototype :

```
long smtp_Quit(long lHandle)
```

Parameters:

lHandle	smtp session handle.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long hndl;

hndl = smtp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   // processing...
   smtp_Quit(hndl);
}
```

smtp_SendMail

Description:

This is the simplest way to send SMTP mail. Use comma to delimit or separate each recipient in the "To", "Cc", or "Bcc" variable and the same rule applies to attached files. The variable eflag is for future use, it should be set to zero.

For valid addresses, please see Mail Address in the appendix.

The following are valid address formats:

"Tony Lee" <hnstech@compuserve.com>

<hnstech@compuserve.com>

hnstech@compuserve.com

Function Prototype :

```
long smtp_SendMail(long lHandle, char *To, char *Cc, char Bcc, char
*Subject, char *Msg, char *attachfiles, long eflag)
```

Parameters:

lHandle	Smtp session handle.
To	To recipients, delimited by comma.
Cc	Carbon Copy recipients, delimited by comma.
Bcc	Blind carbon copy recipients, delimited by comma.
Subject	The subject text
Msg	The mail message.
attachfiles	Attach files, delimited by comma.
eflag	Should be 0. (For future use).

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails.

Example:

```
long hndl;

hndl = stmp_LoginMail("smtp_hostname",  "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   if ( smtp_SendMail(hndl, "70531.1066@compuserve.com",
        "70531.1066@compuserve.com", "", "subject",
        "hello, world", "c:\\junk.txt", 0) == 0 )
   {
      // processing...
   }
   smtp_Quit(hndl);
}
```

smtp_Verify

Description:

Use this function to verify the receiver's mail address. If successful, the full name of the user(if known) and fully specified email address are returned.

Function Prototype :

```
long smtp_Verify(long lHandle, char *user, char *buffer, long blen)
```

Parameters:

lHandle	smtp session handle.
user	a user name.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails. The return buffer string is null terminated.

Example:

```
long hndl;
char buffer[512];

hndl = smtp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   if ( smtp_Verify(hndl, "user name", buffer, 512) >= 0 )
   {
      // processing...
   }
   smtp_Quit(hndl);
}
```

smtp_Version

Description:

This function returns the current version number of SMTPOP16 or SMTPOP32 DLL. The version number returned is an integer, multiplied by 100. Therefore it should be divided by 100 to get the actual version number. For example, The version number returned of 250 means version 2.50.

Function Prototype :

```
long smtp_Version()
```

Parameters:

None.

Return Value:

Long. Actual version number multiplied by 100.

Example:

```
float version;
version = ((float) smtp_Version()) / 100.0;
```

Introduction

This section is intended for Advance STMP developers who are familiar with SMTP protocol RFC 821.

smtp_AddAttachFile

Description:

Attach a file to the mail. You will need to call this function multiple times to add multiple files.

Function Prototype :

```
long smtp_AddAttachFile(long lHandle, char *filename, long filetype,
char *subtype, long encode_type)
```

Parameters:

lHandle	smtp session handle.
filename	The full path of the file to be attached.
filetype	The file type. The default subtypes are in parenthesis. 0 - ASCII text file ("plain") 1 - application binary file. ("octet-stream") 2 - audio file ("basic") 3 - image file ("gif") 4 - message file ("rfc822") 5 - video file ("mpeg")
subtype	Subtype of the file. Leave it empty for default. For more information, please refer to RFC#1521.
encode_type	The encoding scheme. 0 - no encode needed (only for ASCII text file). 1 - Base 64 encoding.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   rc=smtp_InitMail(li_hndl);
   smtp_LastMsg(msg, 512);
   m_log = m_log + msg;  // display the reply message.
   if ( rc == 0 )
   {
      smtp_AddAttachFile(hndl, "c:\\junk.txt", 0, "", 0);
      smtp_AddAttachFile(hndl, "c:\\app.exe", 1, "", 1);
      smtp_AddAttachFile(hndl, "c:\\readme.doc", 1, "", 1);
      rc=smtp_To(li_hndl, To); // Put it in a loop if you have
                          // more than one To, CC, or BCC users.
   OR
```

```
      rc = smtp_Cc(li_hndl, cc);
   OR
      rc = smtp_Bcc(li_hndl, bcc);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   if ( rc == 0 )
   {
      smtp_Subject(li_hndl, "Test mail");
      rc = smtp_MailBodyText(li_hndl, m_mailtext);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   smtp_Quit(li_hndl);
}
```

smtp_Bcc

Description:

Send blind carbon copy user's address to the SMTP mail server. A comma "," must be used to separate each blind carbon copy user. This function can be called more than once in a mail session. For valid addresses, please see Mail Address in the appendix.

Function Prototype :

```
long smtp_bcc(long lHandle, char *bccusers)
```

Parameters:

lHandle	smtp session handle.
bccusers	the bcc recipients email addresses separated by comma's.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   rc=smtp_InitMail(li_hndl);
   smtp_LastMsg(msg, 512);
   m_log = m_log + msg;  // display the reply message.
   if ( rc == 0 )
   {
      rc=smtp_To(li_hndl, To); // Put it in a loop if you have
                          // more than one To, CC, or BCC users.
   OR
      rc = smtp_Cc(li_hndl, cc);
   OR
      rc = smtp_Bcc(li_hndl, bcc);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   if ( rc == 0 )
   {
      smtp_Subject(li_hndl, "Test mail");
      rc = smtp_MailBodyText(li_hndl, m_mailtext);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   smtp_Quit(li_hndl);
```

```
}
```

smtp_Cc

Description:

Send carbon copy user's address to the SMTP mail server. A comma "," must be used to separated each carbon copy user. This function can be called more than once in a mail session. For valid addresses, please see Mail Address in the appendix.

Function Prototype :

```
long smtp_Cc(long lHandle, char *ccusers)
```

Parameters:

lHandle	smtp session handle.
ccusers	the cc recipients email addresses separated by comma's.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   rc=smtp_InitMail(li_hndl);
   smtp_LastMsg(msg, 512);
   m_log = m_log + msg;  // display the reply message.
   if ( rc == 0 )
   {
      rc=smtp_To(li_hndl, To); // Put it in a loop if you have
                          // more than one To, CC, or BCC users.
   OR
      rc = smtp_Cc(li_hndl, cc);
   OR
      rc = smtp_Bcc(li_hndl, bcc);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   if ( rc == 0 )
   {
      smtp_Subject(li_hndl, "Test mail");
      rc = smtp_MailBodyText(li_hndl, m_mailtext);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   smtp_Quit(li_hndl);
```

```
}
```

smtp_CloseData

Description:

Close the Data Channel that was established with smtp_OpenData. The mail data will be sent and completed after the smtp_CloseData is called successfully.

Function Prototype :

```
long smtp_CloseData(long lHandle)
```

Parameters:

lHandle	smtp session handle.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   rc=smtp_InitMail(li_hndl);
   smtp_LastMsg(msg, 512);
   m_log = m_log + msg;  // display the reply message.
   if ( rc == 0 )
   {
      rc=smtp_To(li_hndl, To); // Put it in a loop if you have
                          // more than one To, CC, or BCC users.
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   if ( rc == 0 )
   {
      rc = smtp_OpenData(li_hndl);
      if ( rc == 0 )
      {
         smtp_WriteData(li_hndl, m_mailtext);
      }
      smtp_CloseData(li_hndl);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   smtp_Quit(li_hndl);
}
```

smtp_Date

Description:

Set the mail date. This is an optional function. Use this function to override the system-generated date. RFC 822 specified date format: [weekday,] dd month yyyy hh:mm:ss timezone.

Function Prototype :

```
long smtp_Date(long lHandle, char *date)
```

Parameters:

lHandle	smtp session handle.
date	the mail date.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   rc=smtp_InitMail(li_hndl);
   smtp_LastMsg(msg, 512);
   m_log = m_log + msg;  // display the reply message.
   if ( rc == 0 )
   {
      rc=smtp_To(li_hndl, To); // Put it in a loop if you have
                          // more than one To, CC, or BCC users.
   OR
      rc = smtp_Cc(li_hndl, cc);
   OR
      rc = smtp_Bcc(li_hndl, bcc);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   if ( rc == 0 )
   {
      smtp_Date(li_hndl, "Fri, 6 Jun 1997 10:00:00 CST");
      smtp_Subject(li_hndl, "Test mail");
      rc = smtp_MailBodyText(li_hndl, m_mailtext);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   smtp_Quit(li_hndl);
}
```

smtp_DefaultTimeOut

Description:

This is an optional function. Use this function to override the default timeout value in seconds. This timeout will be the initial time timeout value for any new smtp or pop3 sessions. The factory default value is 60 seconds.

Function Prototype :

```
long smtp_DefaultTimeOut(long lTimeOut)
```

Parameters:

lTimeOut	new timeout value (in seconds) for the smtp session.

Return Value:

Long. Returns the previous timeout value.

Example:

```
long li_hndl, rc;

// set initial default timeout value to 2 minutes.

rc = smtp_DefaultTimeOut(120);

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
    // set timeout to 10 seconds for the current session.
                    rc=smtp_SetTimeOut(li_hndl, 10);
}
```

smtp_DisableLog

Description:

Use this function to disable and stop tracing.

Function Prototype :

```
long smtp_DisableLog()
```

Parameters:

None.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
// Stop the trace
smtp_DisableLog();
```

smtp_EnableLog

Description:

Use this function to specify the trace log file name and start capturing the SMTP or POP3 trace in the log.

Function Prototype :

```
long smtp_EnableLog(char *fname, long options)
```

Parameters:

fname	Log file name.
options	set this to 0.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
// Start trace log into "smtp.log".
smtp_EnableLog("smtp.log", 0);
```

smtp_InitMail

Description:

Initialize a new mail session with the SMTP mail server. This function must be called before starting a new mail session.

Note: DO NOT call this function if you are using smtp_SendMail.

Function Prototype :

```
long smtp_InitMail(long lHandle)
```

Parameters:

lHandle	smtp session handle.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   rc=smtp_InitMail(li_hndl);
   smtp_LastMsg(msg, 512);
   m_log = m_log + msg;  // display the reply message.
   if ( rc == 0 )
   {
      rc=smtp_To(li_hndl, To); // Put it in a loop if you have
                          // more than one To, CC, or BCC users.
   OR
      rc = smtp_Cc(li_hndl, cc);
   OR
      rc = smtp_Bcc(li_hndl, bcc);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   if ( rc == 0 )
   {
      smtp_Subject(li_hndl, "Test mail");
      rc = smtp_MailBodyText(li_hndl, m_mailtext);
      smtp_GetReply(li_hndl, msg, 512);
      m_log = m_log + msg;
   }
   smtp_Quit(li_hndl);
}
```

smtp_MailBodyText

Description:

Use this function to specify the mail's main body text. If you are using multiple mailing functions (i.e. smtp_InitMail, smtp_To, smtp_Cc, ...), make sure this is the last function in your calling sequence. The mail will be finalized and sent if there is no error.

Function Prototype :

```
long smtp_MailBodyText(long lHandle, char *mailmsg)
```

Parameters:

lHandle	smtp session handle.
mailmsg	mail body message text.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   rc=smtp_InitMail(li_hndl);
   smtp_LastMsg(msg, 512);
   m_log = m_log + msg;  // display the reply message.
   if ( rc == 0 )
   {
      rc=smtp_To(li_hndl, To); // Put it in a loop if you have
                          // more than one To, CC, or BCC users.
   OR
      rc = smtp_Cc(li_hndl, cc);
   OR
      rc = smtp_Bcc(li_hndl, bcc);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   if ( rc == 0 )
   {
      smtp_Subject(li_hndl, "Test mail");
      rc = smtp_MailBodyText(li_hndl, m_mailtext);
      smtp_GetReply(li_hndl, msg, 512);
      m_log = m_log + msg;
   }
   smtp_Quit(li_hndl);
}
```

smtp_OpenData

Description:

Establish and initialize the mail data channel. You must be familiar with SMTP protocol to use this function. This function can only be called after the smtp_To function. If it is successfully opened, use smtp_WriteData or smtp_WriteDataRaw to write to mail data buffer. Close the channel with smtp_CloseData function.

Function Prototype :

```
long smtp_OpenData(long lHandle)
```

Parameters:

lHandle	smtp session handle.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   rc=smtp_InitMail(li_hndl);
   smtp_LastMsg(msg, 512);
   m_log = m_log + msg;  // display the reply message.
   if ( rc == 0 )
   {
      rc=smtp_To(li_hndl, To); // Put it in a loop if you have
                          // more than one To, CC, or BCC users.
   OR
      rc = smtp_Cc(li_hndl, cc);
   OR
      rc = smtp_Bcc(li_hndl, bcc);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   if ( rc == 0 )
   {
      rc = smtp_OpenData(li_hndl);
      if ( rc == 0 )
      {
         smtp_WriteData(li_hndl, m_mailtext);
      OR
         smtp_WriteRawData(li_hndl, m_mailtext);
      }
```

```
      smtp_CloseData(li_hndl);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   smtp_Quit(li_hndl);
}
```

smtp_ReplyCode

Description:

This function returns the last SMTP reply code from SMTP server.

Function Prototype :

```
long smtp_ReplyCode()
```

Parameters:

lHandle	smtp session handle.

Return Value:

Long. Returns the last SMTP reply code.

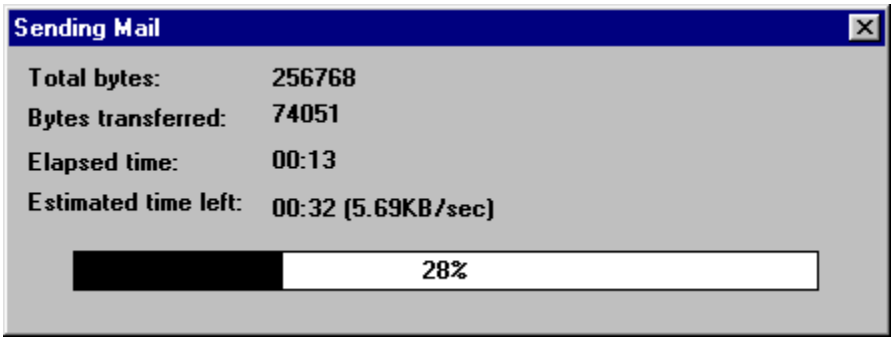
Example:

```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   rc = smtp_ReplyCode(li_hndl);
   // other processing...
}
```

smtp_SetProgressBar

Description:

This function turns the current SMTP session "Progress Bar" dialog box ON or OFF. Default is set by smtpop_DefaultProgressBar function.



Function Prototype :

```
long smtp_SetProgressBar(long lHandle, long onoff_flag)
```

Parameters:

lHandle	smtp session handle.
onoff_flag	1 - On 0 - Off

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
    // set progress bar On for the current session.
    rc smtp_SetProgressBar(li_hndl, 1);
}
```

smtp_SetTimeOut

Description:

This is an optional function. Use this function to override the default timeout value in seconds. This timeout value only applies to the current session.

Function Prototype :

```
long smtp_SetTimeOut(long lHandle, long lTimeOut)
```

Parameters:

lHandle	smtp session handle.
lTimeOut	new timeout value (in seconds) for the smtp session.

Return Value:

Long. Returns the previous timeout value.

Example:

```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
    // set timeout to 10 seconds for the current session.
                    rc=smtp_SetTimeOut(li_hndl, 10);
}
```

smtp_SetTimeZone

Description:

This is an optional function. Use this function to override the system-generated timezone.

Function Prototype :

```
long smtp_SetTimeZone(char *tzone)
```

Parameters:

tzone	The new timezone.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
smtp_SetTimeZone("CST");
```

smtp_Subject

Description:

Set the mail subject.

Function Prototype :

```
long smtp_Subject(long lHandle, char *subject)
```

Parameters:

lHandle	smtp session handle.
subject	the mail subject.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   rc=smtp_InitMail(li_hndl);
   smtp_LastMsg(msg, 512);
   m_log = m_log + msg;  // display the reply message.
   if ( rc == 0 )
   {
      rc=smtp_To(li_hndl, To); // Put it in a loop if you have
                          // more than one To, CC, or BCC users.
   OR
      rc = smtp_Cc(li_hndl, cc);
   OR
      rc = smtp_Bcc(li_hndl, bcc);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   if ( rc == 0 )
   {
      smtp_Subject(li_hndl, "Test mail");
      rc = smtp_MailBodyText(li_hndl, m_mailtext);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   smtp_Quit(li_hndl);
}
```

smtp_To

Description:

Send "TO" user's address to the SMTP mail server. A comma "," must be used to separate each user. This function can be called more than once in a mail session. For valid addresses, please see Mail Address in the appendix.

Function Prototype :

```
long smtp_To(long lHandle, char *tousers)
```

Parameters:

lHandle	smtp session handle.
tousers	the recipients email addresses separated by comma's.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   rc=smtp_InitMail(li_hndl);
   smtp_LastMsg(msg, 512);
   m_log = m_log + msg;  // display the reply message.
   if ( rc == 0 )
   {
      rc=smtp_To(li_hndl, To); // Put it in a loop if you have
                          // more than one To, CC, or BCC users.
   OR
      rc = smtp_Cc(li_hndl, cc);
   OR
      rc = smtp_Bcc(li_hndl, bcc);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   if ( rc == 0 )
   {
      smtp_Subject(li_hndl, "Test mail");
      rc = smtp_MailBodyText(li_hndl, m_mailtext);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   smtp_Quit(li_hndl);
```

```
}
```

smtp_WriteData

Description:

Write mail data into mail data buffer without the new line character. The mail data may contain any of the 128 ASCII character codes and end with newline characters. The data may contain more than one line. Remember to call smtp_OpenData before calling this function. This function may be called more than once. Note: a new line character is "\r\n".

Function Prototype :

```
long smtp_WriteData(long lHandle, char *maildata)
```

Parameters:

lHandle	smtp session handle.
maildata	mail data with new line character.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   rc=smtp_InitMail(li_hndl);
   smtp_LastMsg(msg, 512);
   m_log = m_log + msg;  // display the reply message.
   if ( rc == 0 )
   {
      rc=smtp_To(li_hndl, To); // Put it in a loop if you have
                          // more than one To, CC, or BCC users.
   OR
      rc = smtp_Cc(li_hndl, cc);
   OR
      rc = smtp_Bcc(li_hndl, bcc);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   if ( rc == 0 )
   {
      rc = smtp_OpenData(li_hndl);
      if ( rc == 0 )
      {
         smtp_WriteData(li_hndl, m_mailtext);
      OR
```

```
            smtp_WriteRawData(li_hndl, m_mailtext);
         }
         smtp_CloseData(li_hndl);
         smtp_LastMsg(msg, 512);
         m_log = m_log + msg;
      }
      smtp_Quit(li_hndl);
   }
```

smtp_WriteRawData

Description:

Write mail data into mail data buffer with the new line character. Therefore, "\r\n" will be appended to every smtp_WriteRawData call. The mail data may contain any of the 128 ASCII character codes. Remember to call smtp_OpenData before calling this function. This function may be called more than once.

Function Prototype :

```
long smtp_WriteRawData(long lHandle, char *maildata)
```

Parameters:

lHandle	smtp session handle.
maildata	mail data without new line character.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

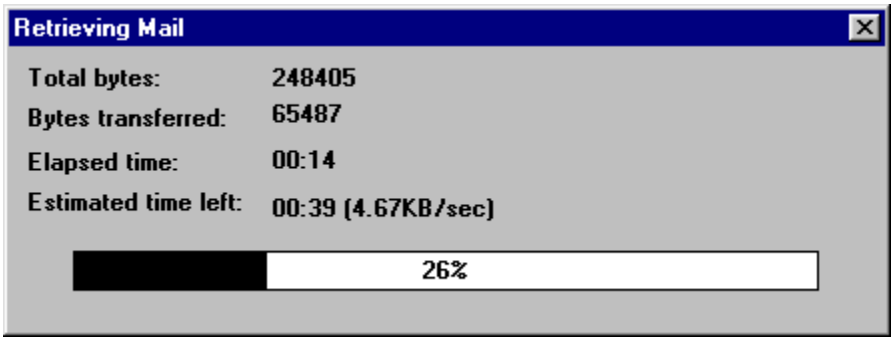
```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   rc=smtp_InitMail(li_hndl);
   smtp_LastMsg(msg, 512);
   m_log = m_log + msg;  // display the reply message.
   if ( rc == 0 )
   {
      rc=smtp_To(li_hndl, To); // Put it in a loop if you have
                          // more than one To, CC, or BCC users.
   OR
      rc = smtp_Cc(li_hndl, cc);
   OR
      rc = smtp_Bcc(li_hndl, bcc);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   if ( rc == 0 )
   {
      rc = smtp_OpenData(li_hndl);
      if ( rc == 0 )
      {
         smtp_WriteData(li_hndl, m_mailtext);
      OR
         smtp_WriteRawData(li_hndl, m_mailtext);
```

```
      }
      smtp_CloseData(li_hndl);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   smtp_Quit(li_hndl);
}
```

smtpop_DefaultProgressBar

Description:

Set the default "Progress Bar" dialog box ON or OFF for all new SMTP or POP3 sessions. Default is OFF.



Function Prototype :

```
long smtpop_DefaultProgressBar(long onoff_flag)
```

Parameters:

onoff_flag	1 - On
	0 - Off

Return Value:

Long. Returns previous value of the default "Progress Bar" state.

Example:

```
long li_hndl, rc;
// set default progress bar On for all new sessions.
rc = smtp_DefaultProgressBar(1);
```

SMTP C++ Example

Description:

The following examples illustrates the use of Advance SMTP functions in C++ code. This is not a complete or functional code. Not all the variables are declared in the example.

Example:

```
long li_hndl, rc;

li_hndl = stmp_LoginMail("smtp_hostname", "myname", "myid", "mydomain");
if ( hndl > 0 )
{
   rc=smtp_InitMail(li_hndl);
   smtp_LastMsg(msg, 512);
   m_log = m_log + msg;  // display the reply message.
   if ( rc == 0 )
   {
      rc=smtp_To(li_hndl, To); // Put it in a loop if you have
                          // more than one To, CC, or BCC users.
   OR
      rc = smtp_Cc(li_hndl, cc);
   OR
      rc = smtp_Bcc(li_hndl, bcc);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   if ( rc == 0 )
   {
      smtp_Subject(li_hndl, "Test mail");
      rc = smtp_MailBodyText(li_hndl, m_mailtext);
      smtp_LastMsg(msg, 512);
      m_log = m_log + msg;
   }
   smtp_Quit(li_hndl);
}
```

Introduction

This section describes the basic POP3 functions. For each function, it gives a brief description of the function, the function prototype, and a simple example on how to use it. The function prototype describes, in C syntax, the calling sequence and parameters needed for the function to operate properly.

Please refer to the appendix for information on how to obtain function prototypes for other programming languages.

This manual sometime refers to a mail part as attached mail part or attached file. They are synonymous. The pop3_GetAttachCount actually returns the number of attached mail parts in a mail and not necessarily the number of attachments. To obtain the number of attachments, you will have to loop through the number of mail parts and decides whether the mail part is an attachment using function like pop3_GetDispositionType, pop3_GetDispositionFname, or pop3_GetAttachedName.

pop3_DeleteMail

Description:

Delete a mail message from the POP3 mailbox. The POP3 server marks the message as deleted. The POP3 server does not actually delete the message until the POP3 session is terminated with a pop3_Quit function call.

Function Prototype :

```
long pop3_DeleteMail(long lHandle, long MsgNum)
```

Parameters:

lHandle	pop3 session handle.
MsgNum	Mail message number to be deleted.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

pop3_GetAttachCount

Description:

This function returns the number of attached mail parts in a mail i.e. the number of multiple parts. There is at least one part for every mail.

Function Prototype :

```
long pop3_GetAttachCount(long lHandle)
```

Parameters:

lHandle	pop3 session handle.

Return Value:

Long. Returns number of multiple parts, -1 if it fails.

Example:

```
long hndl, startpos;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      if ( pop3_GetAttachCount(hndl) > 0 )
      {
         pop3_GetAttachedName(hndl, 1, buffer, 512, 0);
         // other processing....
      }
   }
   pop3_Quit(hndl);
}
```

pop3_GetAttachedName

Description:

Get the name of the content-type from a given attached mail part, if any. The mail part number is one-based. Mail part number one is normally the mail body if the mail contains multiple parts.

Function Prototype :

long pop3_GetAttachedName(long lHandle, long AttNum, char *buffer, long blen, long startpos)

Parameters:

lHandle	pop3 session handle.
AttNum	Attached file number.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.
startpos	The starting position of the filename to start copying. The startpos is zero-based.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl, startpos;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 && pop3_GetAttachCount(hndl) >
0 )
   {
       pop3_GetAttachedName(hndl, 1, buffer, 512, 0);
     OR
       startpos = 0;
       while ( pop3_GetAttachName(hndl, 1, buffer, 512, startpos) > 0 )
       {
          startpos = startpos + strlen(buffer);
       }
   }
   pop3_Quit(hndl);
}
```

pop3_GetAttachedType

Description:

Get the type and subtype of a content-type from an attached mail part, if any. An example of a file content type and subtype is "text/plain". The mail part number is one-based. Mail part number one is normally the mail body if the mail contains multiple parts. Empty string indicates none found.

Function Prototype :

```
long pop3_GetAttachedType(long lHandle, long AttNum, char *buffer, long blen, long startpos)
```

Parameters:

lHandle	pop3 session handle.
AttNum	Attached file number.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.
startpos	The starting position of the file content type and subtype to start copying. The startpos is zero-based.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl, startpos;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 && pop3_GetAttachCount(hndl) > 0 )
   {
       pop3_GetAttachedName(hndl, 1, buffer, 512, 0);
       // processing
       pop3_GetAttachType(hndl, 1, buffer, 512, 0)
     OR
       startpos = 0;
       while ( pop3_GetAttachType(hndl, 1, buffer, 512, startpos) > 0 )
       {
          startpos = startpos + strlen(buffer);
       }
```

```
    }
    pop3_Quit(hndl);
}
```

pop3_GetDispositionFname

Description:

Get the filename parameter from the content-disposition header of an attached mail part, if any. The mail part number is one-based (i.e. start from one). Empty string indicates none found.

Function Prototype :

```
long pop3_GetDispositionFname(long lHandle, long PartNum, char *buffer,
long blen, long startpos)
```

Parameters:

lHandle	pop3 session handle.
PartNum	Mail attached part number.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.
startpos	The starting position of the Disposition filename to start copying. The startpos is zero-based.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl, startpos;
char buffer[512], type[12];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      // Get the header type of "content-disposition"
      // of mail part number 1
      // e.g. if header = Content-Disposition: attachment;
      // filename="Data.bin" then the following call will
      // return attachment.
      pop3_GetDispositionType(hndl, 1, type, 12, 0);
      if ( strcmp(type, "attachment") == 0 )
      {
         // And the following call will return Data.bin.
          pop3_GetDispositionFame(hndl, 1, buffer, 512, 0);
         // processing...
      }
   }
```

```
    pop3_Quit(hndl);
}
```

pop3_GetDispositionType

Description:

Get the disposition type from the content-disposition header of an attached mail part. The current standard content-disposition types are "inline" and "attachment". The mail part number is one-based (i.e. start from one). Empty string indicates none found.

Function Prototype :

```
long pop3_GetDispositionType(long lHandle, long PartNum, char *buffer,
long blen, long startpos)
```

Parameters:

lHandle	pop3 session handle.
PartNum	Attached mail part number.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.
startpos	The starting position of the Disposition Type to start copying. The startpos is zero-based.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl, startpos;
char buffer[512], type[12];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      // Get the header type of "content-disposition"
      // of mail part number 1
      // e.g. if header = Content-Disposition: attachment;
      // filename="Data.bin" then the following call will
      // return attachment.
      pop3_GetDispositionType(hndl, 1, type, 12, 0);
      if ( strcmp(type, "attachment") == 0 )
      {
         // And the following call will return Data.bin.
          pop3_GetDispositionFame(hndl, 1, buffer, 512, 0);
         // processing...
      }
```

```
    }
    pop3_Quit(hndl);
}
```

pop3_GetHeaderField

Description:

This function returns the header field (i.e. the entire line) of a given header (case insensitive) from an attached mail part. The mail part number is one-based (i.e. start from one). Empty string indicates none found.

Function Prototype :

```
long pop3_GetHeaderField(long lHandle, long PartNum, char *header, char *buffer, long blen, long startpos)
```

Parameters:

lHandle	pop3 session handle.
PartNum	Attached mail part number.
Header	Header to retrieve.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.
startpos	The starting position of the Header Field to start copying. The startpos is zero-based.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      // Get the header type of "content-disposition"
      // of mail part number 1
      // e.g. if header = Content-Disposition: attachment;
      // filename="Data.bin" then the following call will
      // return Content-Disposition: attachment; filename="Data.bin".
      pop3_GetHeaderField(hndl, 1, "content-disposition", buffer, 512,
0);
      // processing...
   }
   pop3_Quit(hndl);
}
```

pop3_GetHeaderType

Description:

This function returns the header type of a given header (case insensitive) from an attached mail part. (e.g. the header type of "content-disposition" is "inline" or "attachment"). The mail part number is one-based (i.e. start from one). Empty string indicates none found.

Function Prototype :

```
long pop3_GetHeaderType(long lHandle, long PartNum, char *header, char
*buffer, long blen, long startpos)
```

Parameters:

lHandle	pop3 session handle.
PartNum	Attached mail part number.
Header	Header to retrieve.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.
startpos	The starting position of the Header Field to start copying. The startpos is zero-based.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      // Get the header type of "content-disposition"
      // of mail part number 1
      // e.g. if header = Content-Disposition: attachment;
      // filename="Data.bin" then the following call will
      // return attachment.
      pop3_GetHeaderType(hndl, 1, "content-disposition", buffer, 512,
0);
      if ( strcmp(buffer, "attachment") == 0 )
      {
         // processing...
      }
```

```
    }
    pop3_Quit(hndl);
}
```

pop3_GetMailCc

Description:

Get the mail's Carbon Copy list. The list is delimited with "\r\n" if multiple lines are received. There may be multiple addresses per line delimited by commas. Example: "H&S Tech" <hnstech@compuserve.com>, joe@company.com. Refer to RFC#822 for full details. This function is only valid after a successful call to pop3_RetrieveMail, pop3_QuickScanMail, or pop3_QuickScanFile function.

Function Prototype :

```
long pop3_GetMailCc(long lHandle, char *buffer, long blen, long
startpos)
```

Parameters:

lHandle	pop3 session handle.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.
startpos	The starting position of the mail "Cc" text to start copying. The startpos is zero-based.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      pop3_GetMailCc(hndl, buffer, 512, 0);
    OR
         startpos = 0;
      while ( pop3_GetMailCc(hndl, buffer, 512, startpos) > 0 )
         {
         startpos = startpos + strlen(buffer);
         }
   }
   pop3_Quit(hndl);
}
```

pop3_GetMailDate

Description:

Get the mail's date. This function is only valid after a successful call to pop3_RetrieveMail, pop3_QuickScanMail, or pop3_QuickScanFile function.

Function Prototype :

```
long pop3_GetMailDate(long lHandle, char *buffer, long blen, long startpos)
```

Parameters:

lHandle	pop3 session handle.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.
startpos	The starting position of the mail date text to start copying. The startpos is zero-based.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl, startpos;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      pop3_GetMailDate(hndl, buffer, 512, 0);
    OR
         startpos = 0;
      while ( pop3_GetMailDate(hndl, buffer, 512, startpos) > 0 )
         {
         startpos = startpos + strlen(buffer);
         }
   }
   pop3_Quit(hndl);
}
```

pop3_GetMailFrom

Description:

Get the mail from sender's address. This function is only valid after a successful call to pop3_RetrieveMail, pop3_QuickScanMail, or pop3_QuickScanFile function.

Function Prototype :

```
long pop3_GetMailFrom(long lHandle, char *buffer, long blen, long startpos)
```

Parameters:

lHandle	pop3 session handle.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.
startpos	The starting position of the mail from text to start copying. The startpos is zero-based.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl, startpos;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      pop3_GetMailFrom(hndl, buffer, 512, 0);
     OR
         startpos = 0;
      while (pop3_GetMailFrom(hndl, buffer, 512, startpos) > 0 )
      {
         startpos = startpos + strlen(buffer);
      }
   }
   pop3_Quit(hndl);
}
```

pop3_GetMailHeader

Description:

Get the mail header text. This function is only valid after a successful call to pop3_RetrieveMail, pop3_QuickScanMail, or pop3_QuickScanFile function.

Function Prototype :

```
long pop3_GetMailHeader(long lHandle, char *buffer, long blen, long startpos)
```

Parameters:

lHandle	pop3 session handle.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.
startpos	The starting position of the mail header text to start copying. The startpos is zero-based.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl, startpos, cc;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      startpos = 0;
      cc = pop3_GetMailHeader(hndl, buffer, 512, startpos);
      while ( cc > 0 )
      {
         startpos = startpos + strlen(buffer);
         cc = pop3_GetMailHeader(hndl, buffer, 512, startpos);
      }
   }
   pop3_Quit(hndl);
}
```

pop3_GetMailSize

Description:

Get the mail message size in bytes.

Function Prototype :

```
long pop3_GetMailSize(long lHandle, long MsgNumber)
```

Parameters:

lHandle	pop3 session handle.
MsgNumber	Mail message number.

Return Value:

Long. Returns the size of the mail message in bytes if it succeeds, -1 if it fails.

Example:

```
long hndl, lsize;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   lsize = pop3_GetMailSize(hndl, 1);
   pop3_Quit(hndl);
}
```

pop3_GetMailSubject

Description:

Get the mail subject. This function is only valid after a successful call to pop3_RetrieveMail, pop3_QuickScanMail, or pop3_QuickScanFile function.

Function Prototype :

```
long pop3_GetMailSubject(long lHandle, char *buffer, long blen, long startpos)
```

Parameters:

lHandle	pop3 session handle.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.
startpos	The starting position of the mail subject text to start copying. The startpos is zero-based.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl, startpos;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      pop3_GetMailSubject(hndl, buffer, 512, 0);
    OR
         startpos = 0;
      while( pop3_GetMailSubject(hndl, buffer, 512, startpos) > 0 )
      {
         startpos = startpos + strlen(buffer);
      }
   }
   pop3_Quit(hndl);
}
```

pop3_GetMailTo

Description:

Get the mail's "To" list. The list is delimited with "\r\n" if multiple lines are received. There may be multiple addresses per line delimited by commas. Example: "H&S Tech" <hnstech@compuserve.com>, joe@company.com. Refer to RFC#822 for full details. This function is only valid after a successful call to pop3_RetrieveMail, pop3_QuickScanMail, or pop3_QuickScanFile function.

Function Prototype :

```
long pop3_GetMailTo(long lHandle, char *buffer, long blen, long
startpos)
```

Parameters:

lHandle	pop3 session handle.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.
startpos	The starting position of the mail "To" text to start copying. The startpos is zero-based.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl, startpos, cc;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      startpos = 0;
      cc = pop3_GetMailTo(hndl, buffer, 512, startpos);
      while ( cc > 0 )
      {
         startpos = startpos + strlen(buffer);
         cc = pop3_GetMailTo(hndl, buffer, 512, startpos);
      }
   }
   pop3_Quit(hndl);
}
```

pop3_GetParmValue

Description:

This function returns the parameter value of a given parameter and header (case insensitive) from an attached mail part. The mail part number is one-based (i.e. start from one). Empty string indicates none found.

Function Prototype :

```
long pop3_GetParmValue(long lHandle, long PartNum, char *header, char
*parm, char *buffer, long blen, long startpos)
```

Parameters:

lHandle	pop3 session handle.
PartNum	Attached mail part number.
Header	Header to retrieve.
Parm	Parameter field name.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.
startpos	The starting position of the Header Field to start copying. The startpos is zero-based.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl;
char buffer[512], fname[80];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      // Get the header type of "content-disposition" of mail
      // part number 1
      pop3_GetHeaderType(hndl, 1, "content-disposition", buffer, 512,
0);
      if ( strcmp(buffer, "attachment") == 0 )
      {
         // Get the parameter value of "filename" from
         // "content-disposition" header type of mail
         // part number 1
         // e.g. if header = Content-Disposition: attachment;
```

```
            // filename="Data.bin" then the following call will
            // return Data.bin.
            pop_GetParmValue(hndl, 1, "content-disposition", "filename",
               fname, 512, 0);
            // processing...
         }
      }
      pop3_Quit(hndl);
   }
```

pop3_GetPartHeaders

Description:

Get an attached mail part's headers. This function returns all the headers of a mail part. The mail part number is one-based (i.e. start from one). Empty string indicates none found.

Function Prototype :

```
long pop3_GetPartHeaders(long lHandle, long PartNum, char *buffer, long blen, long startpos)
```

Parameters:

lHandle	pop3 session handle.
PartNum	Attached mail part number.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.
startpos	The starting position of the Part Headers to start copying. The startpos is zero-based.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      // Get the mail part headers infomation of
      // part number 1.
      pop3_GetPartHeaders(hndl, 1, buffer, 512, 0);
      // processing...
   }
   pop3_Quit(hndl);
}
```

pop3_GetReply

Description:

This function retrieves the current pop3 mailbox server replies from the reply buffer. The return buffer size must be large enough to hold the results.

Function Prototype :

```
long pop3_GetReply(long lHandle, char *buffer, long blen)
```

Parameters:

lHandle	pop3 session handle.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   pop3_GetReply(hndl, buffer, 512);
   pop3_Quit(hndl);
}
```

pop3_GetResults

Description:

Get the mail result after a successful call to pop3_ListMails function. The return buffer size must be large enough to hold the results.

Function Prototype :

```
long pop3_GetResults(long lHandle, char *buffer, long blen)
```

Parameters:

lHandle	pop3 session handle.
buffer	Pointer to user-specified buffer for receiving returned results.
blen	The maximum size of the buffer.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_ListMails(hndl) > 0 )
   {
      pop3_GetReply(hndl, buffer, 512);
   }
   pop3_Quit(hndl);
}
```

pop3_LastMailRead

Description:

Get the last highest mail message number that had been read.

Function Prototype :

```
long pop3_LastMailRead(long lHandle)
```

Parameters:

lHandle	pop3 session handle.

Return Value:

Long. Returns last highest read mail message number if it succeeds and -1 if it fails.

Example:

```
long hndl, rc;

hndl = pop3_LoginMail("smtp_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   rc = pop3_LastMailRead(hndl);
   // processing...
   pop3_Quit(hndl);
}
```

pop3_LastMsg

Description:

This function retrieves the last function message into a user-specified buffer. The message can be of information, warning, or error. It may not necessarily be the message from the server. Please refer to Error Messages for detail of the messages. The return string in the buffer is null terminated.

Function Prototype :

```
long pop3_LastMsg(char *buffer, long blen)
```

Parameters:

buffer	Pointer to user-specified buffer for receiving message data.
blen	The maximum size of the buffer.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails, or the number of bytes that was truncated because the buffer size is too small. The return buffer string is null terminated.

Example:

```
long hndl;
char msg[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   // processing...
   pop3_Quit(hndl);
}
else
   pop3_LastMsg(msg, 512);
```

pop3_ListMails

Description:

List all the mails in the POP3 mailbox. If successful, use pop3_GetResults to retrieve the list.

Function Prototype :

```
long pop3_ListMails(long lHandle)
```

Parameters:

lHandle	pop3 session handle.

Return Value:

Long. Returns the number of mail messages in the mailbox if it succeeds and -1 if it fails.

Example:

```
long hndl;
char buffer[1024];

hndl = pop3_LoginMail("smtp_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_ListMails(hndl) > 0 )
      pop3_GetResults(hndl, buffer, 1024);
   // processing...
   pop3_Quit(hndl);
}
```

pop3_Login

Description:

Establish a POP3 mail session with the POP3 mailbox server (host) on the well-known POP3 port.

Function Prototype :

```
long pop3_Login(long hndl, char *pop3server, char *username, char *password)
```

Parameters:

hndl	Pop3 session handle.
pop3server	Internet host name or IP address of your pop3 mailbox server
username	Your mail user id.
password	Your mail id's password.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long hndl;
hndl = pop3_NewHandle();
if ( hndl > 0 )
{

   if ( pop3_Login(hndl, "smtp_hostname", "myid", "mypassword") == 0 )
   {
       // processing...
   }
   pop3_Quit(hndl);
}
```

pop3_LoginMail

Description:

Establish a POP3 session with the POP3 mailbox server (host) on the well-known POP3 port. This is the same as calling pop3_NewHandle then pop3_Login.

Function Prototype :

```
long pop3_LoginMail(char *pop3server, char *username, char *password)
```

Parameters:

pop3server	internet host name or IP address of your pop3 mailbox server
username	Your mail user id.
password	Your mail id's password.

Return Value:

Long. Returns a handle to the POP3 session if it succeeds and -1 if it fails. The valid handle is always greater than 0.

Example:

```
long hndl;

hndl = pop3_LoginMail("smtp_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   // processing...
   pop3_Quit(hndl);
}
```

pop3_LoginPort

Description:

Establish a POP3 mail session with the POP3 mailbox server (host) on a given POP3 port.

Function Prototype :

```
long pop3_LoginPort(long hndl, char *pop3server, long port, char *username, char *password)
```

Parameters:

hndl	Pop3 session handle.
pop3server	Internet host name or IP address of your pop3 mailbox server
port	The POP3 mail server port.
username	Your mail user id.
password	Your mail id's password.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long hndl;
hndl = pop3_NewHandle();
if ( hndl > 0 )
{
   if ( pop3_LoginPort(hndl, "smtp_hostname", 1234, "myid",
"mypassword") == 0 )
   {
       // processing...
   }
   pop3_Quit(hndl);
}
```

pop3_MailStatus

Description:

Get POP3 mailbox status. Use pop3_GetReply() function to obtain the reply from mailbox server.

Function Prototype :

```
long pop3_MailStatus(long lHandle)
```

Parameters:

lHandle	pop3 session handle.

Return Value:

Long. Returns the number of mail messages in the mailbox if it succeeds and -1 if it fails.

Example:

```
long hndl;
char buffer[1024];

hndl = pop3_LoginMail("smtp_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   pop3_MailStatus(hndl);
   pop3_GetReply(hndl, buffer, 1024);
   // processing...
   pop3_Quit(hndl);
}
```

pop3_Noop

Description:

Send a "no operation" command to the pop3 mailbox server. This function does not affect any parameters or previously called functions. The pop3 mailbox server does not perform any action other than acknowledging the call.

Function Prototype :

```
long pop3_Noop(long lHandle)
```

Parameters:

lHandle	pop3 session handle.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long hndl;
char buffer[1024];

hndl = pop3_LoginMail("smtp_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   pop3_Noop(hndl);
   // processing...
   pop3_Quit(hndl);
}
```

pop3_NewHandle

Description:

Obtain a new session handle. This function does not logon to POP3 server. You will need to call pop3_Login to establish a POP3 mail session (see also pop3_LoginMail).

This function allows you to read mails that are already retrieved into files (see pop3_RetrieveIntoFile and pop3_OpenMailFile) without having to establish a POP3 mail session.

Function Prototype :

```
long pop3_NewHandle()
```

Parameters:

None.

Return Value:

Long. Returns a handle to the POP3 session if it succeeds and -1 if it fails. The valid handle is always greater than 0.

Example:

```
long hndl;
hndl = pop3_NewHandle();
if ( hndl > 0 )
{
   if ( pop3_OpenMailFile(hndl, "mail.txt") == 0 )
   {
      // processing
   }
OR...
   if ( pop3_Login(hndl, "smtp_hostname", "myid", "mypassword") == 0 )
   {
       // processing...
   }
   pop3_Quit(hndl);
}
```

pop3_OpenAttached

Description:

Open an attached mail part for reading. This function also closes the previously opened attached mail part or mail body text. Only one attached mail part can be opened at a time. Use pop3_ReadAttached to read the part.

Function Prototype :

```
long pop3_OpenAttached(long lHandle, long AttNum)
```

Parameters:

lHandle	pop3 session handle.
AttNum	The attached file number to open.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails.

Example:

```
long hndl;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      if ( pop3_GetAttachCount(hndl) > 0 )
      {
          if ( pop3_OpenAttached(hndl, 1) == 0 )
          {
              while((n = pop3_ReadAttached(hndl, buffer, 512)) > 0 )
              {
               buffer[n] = '\0';
               // other processing.....
            }
         }
      }
   }
   pop3_Quit(hndl);
}
```

pop3_OpenReadBodyText

Description:

Open and initialize the mail body for reading. This function also closes the previous opened file attachment. Only one attached file can be opened at a time. Use pop3_ReadBodyText to read the mail body text.

This is equivalent to doing pop3_OpenAttached(handle, 1).

Function Prototype :

```
long pop3_OpenReadBodyText(long lHandle)
```

Parameters:

lHandle	pop3 session handle.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails.

Example:

```
long hndl;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      if ( pop3_OpenReadBodyText(hndl) == 0 )
      {
          while((n = pop3_ReadBodyText(hndl, buffer, 512)) > 0 )
          {
              buffer[n] = '\0';
              // other processing.....
          }
      }
   }
   pop3_Quit(hndl);
}
```

pop3_OpenMailFile

Description:

Open a Mail file, which must be MIME 1.0 compliance. Once opened, you can use any of the Read Mail functions to retrieve mail content.

Function Prototype :

```
long pop3_OpenMailFile(long lHandle, char *filename)
```

Parameters:

lHandle	pop3 session handle.
filename	File name.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails.

Example:

```
long hndl;
char buffer[512];

hndl = pop3_NewHandle();
OR
hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_OpenMailFile(hndl, "c:\mail1.txt") == 0 )
   {
      pop3_GetMailFrom(hndl, buffer, 512, 0);
....
      if ( pop3_GetAttachCount(hndl) > 0 )
     {
        pop3_GetAttachedName(hndl, buffer, 512, 0);
....
     }
     // other processing....
   }
   pop3_Quit(hndl);
}
```

pop3_QuickScanFile

Description:

Open and scan the mail header of a given mail file, which must be MIME 1.0 compliance. If successful, use the GET functions (i.e. functions that start with pop3_Get) to retrieve the mail information. It returns the total number of bytes in the mail header.

Function Prototype :

```
long pop3_QuickScanFile(long lHandle, char *filename)
```

Parameters:

lHandle	pop3 session handle.
filename	File name.

Return Value:

Long. Returns > 0 if it succeeds and -1 if it fails.

Example:

```
long hndl;
char buffer[512];

hndl = pop3_NewHandle();
OR
hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_QuickScanFile(hndl, "c:\mail1.txt") == 0 )
   {
      pop3_GetMailFrom(hndl, buffer, 512, 0);
....
     // other processing....
   }
   pop3_Quit(hndl);
}
```

pop3_QuickScanMail

Description:

Retrieve just the header information of a given mail message number from the POP3 mailbox. If successful, use the GET functions (i.e. functions that start with pop3_Get) to retrieve the mail information. It returns the total number of bytes in the mail header.

Function Prototype :

```
long pop3_QuickScanFile(long lHandle, long MsgNum)
```

Parameters:

lHandle	pop3 session handle.
MsgNum	Mail message number.

Return Value:

Long. Returns > 0 if it succeeds and -1 if it fails.

Example:

```
long hndl;
char buffer[1024];

hndl = pop3_LoginMail("smtp_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_QuickScanMail(hndl, 1) > 0 )
   {
      pop3_GetMailFrom(hndl, buffer, 1024, 0);
      OR
      pop3_GetMailHeader(hndl, buffer, 1024, 0);
      OR
      pop3_GetMailSubject(hndl, buffer, 1024, 0);
      // processing...
   }
   pop3_Quit(hndl);
}
```

pop3_Quit

Description:

Quit and disconnect from POP3 mailbox server, if connected, and release POP3 session control block. This also force the POP3 server to delete all mails marked as deleted.

Function Prototype :

```
long pop3_Quit(long lHandle)
```

Parameters:

lHandle	pop3 session handle.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long hndl;
char buffer[1024];

hndl = pop3_LoginMail("smtp_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   // processing...
   pop3_Quit(hndl);
}
```

pop3_ReadAttached

Description:

Read decoded data into the user-supplied buffer from attached mail part. The number of bytes read can be less than the number of bytes requested and zero byte indicates the end of the mail part.

Function Prototype :

```
long pop3_ReadAttached(long lHandle, char *buffer, long nbytes)
```

Parameters:

lHandle	pop3 session handle.
buffer	Pointer to the user-supplied buffer that is used to receive the data read.
nbytes	The maximum number of bytes to be read from the attached mail part.

Return Value:

Long. Returns the number of bytes read into the buffer if it succeeds, 0 if it reaches the end of mail part, and -1 if it fails. The return buffer is not null terminated.

Example:

```
long hndl;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 && pop3_GetAttachCount(hndl) >
0 )
   {
       if ( pop3_OpenAttached(hndl, 1) == 0 )
       {
           while((n = pop3_ReadAttached(hndl, buffer, 512)) > 0 )
           {
               buffer[n] = '\0';
               // other processing.....
           }
       }
   }
   pop3_Quit(hndl);
}
```

pop3_ReadBodyText

Description:

Read decoded mail body text into the user-supplied buffer. The number of bytes read can be less than the number of bytes requested and zero byte indicates the end of text.

Function Prototype :

```
long pop3_ReadBodyText(long lHandle, char *buffer, long nbytes)
```

Parameters:

lHandle	pop3 session handle.
buffer	Pointer to the user-supplied buffer that is used to receive the data read.
nbytes	The maximum number of bytes to be read from the mail body.

Return Value:

Long. Returns the number of bytes read into the buffer if it succeeds, 0 if it reaches the end of mail body, and -1 if it fails. The return buffer is not null terminated.

Example:

```
long hndl;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      if ( pop3_OpenReadBodyText(hndl) == 0 )
      {
          while((n = pop3_ReadBodyText(hndl, buffer, 512)) > 0 )
          {
              buffer[n] = '\0';
              // other processing.....
          }
      }
   }
   pop3_Quit(hndl);
}
```

pop3_ResultsSize

Description:

Query the size (in bytes) of the results. Use this function to determine the buffer size required to retrieve the results.

Function Prototype :

```
long pop3_ResultsSize(long lHandle)
```

Parameters:

lHandle	pop3 session handle.

Return Value:

Long. Returns number of bytes if it succeeds and -1 if it fails.

Example:

```
long hndl;
char *buffer;

hndl = pop3_LoginMail("smtp_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   len = pop3_ResultsSize(hndl);
   buffer = (char *) malloc(len+1);
   pop3_GetResults(hndl, buffer, len);
   // processing...
   pop3_Quit(hndl);
   free(buffer);
}
```

pop3_ResetMail

Description:

Reset the mailbox to its original state. That is, unmarked all mail messages that were marked as deleted and reset the highest accessed mail number.

Function Prototype :

```
long pop3_ResetMail(long lHandle)
```

Parameters:

lHandle	pop3 session handle.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

pop3_RetrieveMail

Description:

Retrieve mail from the POP3 mailbox. The mail message number, MsgNum, determines the mail message. If successful, use the "Get" functions (i.e. functions that start with pop3_Get) to retrieve the result. It returns the total number of bytes in the mail.

Function Prototype :

```
long pop3_RetrieveMail(long lHandle, long MsgNum)
```

Parameters:

lHandle	pop3 session handle.
MsgNum	Mail message number.

Return Value:

Long. Returns > 0 if it succeeds and -1 if it fails.

Example:

```
long hndl;
char buffer[1024];

hndl = pop3_LoginMail("smtp_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      pop3_GetResults(hndl, buffer, 1024);
      OR
      pop3_GetMailHeader(hndl, buffer, 1024. 0);
      // processing...
   }
   pop3_Quit(hndl);
}
```

pop3_RetrieveIntoFile

Description:

Retrieve mail from the POP3 mailbox and save it into a file. The mail message number, MsgNum, determines the mail message. If successful, use pop3_OpenMailFile to process the mail.

Function Prototype :

```
long pop3_RetrieveIntoFile(long lHandle, long MsgNum, char *filename,
long owflag)
```

Parameters:

lHandle	pop3 session handle.
MsgNum	Mail message number to be retrieved.
filename	The file name where to store the retrieved mail.
owflag	The override flag. 0 - not to override an existing file. Will prompt if file exists. 1 - override if file exists.

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long hndl;
char buffer[1024];

hndl = pop3_LoginMail("smtp_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   pop3_RetrieveIntoFile(hndl, 1, "c:\\temp\\mail.txt", 1);
   pop3_OpenMailFile(hndl, "c:\\temp\\mail.txt");
   // processing...
   pop3_Quit(hndl);
}
```

pop3_SaveAttachedTo

Description:

Save the attached mail part, if any, to a file. The attached mail part number is one-based. Attached mail part one is normally the mail body text if the mail contains multiple attached parts.

Function Prototype :

```
long pop3_SaveAttachedTo(long lHandle, long AttNum, char *filename, owflag)
```

Parameters:

lHandle	pop3 session handle.
AttNum	Attached file number.
filename	The filename to save to. Empty string will default to the attached content-type's name.
owflag	Override flag. 0 - do not override file if file exists. Prompt user for new file. 1 - always override file.

Return Value:

Long. Returns 0 if it succeeds, -1 if it fails.

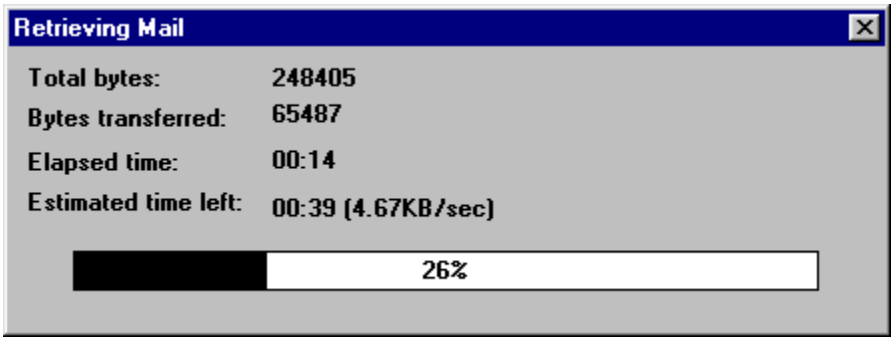
Example:

```
long hndl;
char buffer[512];

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
   if ( pop3_RetrieveMail(hndl, 1) > 0 )
   {
      if ( pop3_GetAttachCount(hndl) > 0 )
      {
         for(i=1; i <= pop3_GetAttachCount(hndl); i++)
         {
          pop3_SaveAttachedTo(hndl, i, "", 0);
        OR
          pop3_SaveAttachedTo(hndl, i, "c:\temp.txt", 1);
         }
      }
   }
   pop3_Quit(hndl);
}
```

pop3_SetProgressBar

Description:

This function turns current POP3 session "Progress Bar" dialog box ON or OFF. Default is set by smtpop_DefaultProgressBar function.



Function Prototype :

```
long pop3_SetProgressBar(long lHandle, long onoff_flag)
```

Parameters:

lHandle	pop3 session handle.
onoff_flag	1 - On 0 - Off

Return Value:

Long. Returns 0 if it succeeds and -1 if it fails.

Example:

```
long hndl, rc;

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
    // set progress bar On for the current session.
    rc = pop3_SetProgressBar(li_hndl, 1);
}
```

pop3_SetTimeOut

Description:

This is an optional function. Use this function to override the default timeout value in seconds. This timeout value only applies to the current session.

Function Prototype :

```
long pop3_SetTimeOut(long lHandle, long lTimeOut)
```

Parameters:

lHandle	pop3 session handle.
lTimeOut	new timeout value (in seconds) for the pop3 session.

Return Value:

Long. Returns the previous timeout value.

Example:

```
long hndl, rc;

hndl = pop3_LoginMail("pop3_hostname", "myid", "mypassword");
if ( hndl > 0 )
{
    // set timeout to 10 seconds for the current session.
                    rc=pop3_SetTimeOut(hndl, 10);
}
```

Mail Address

The valid mail addresses are as follow:

"user name" <mailid@company.com>

"user name" mailid@company.com

<mailid@company.com>

mailid@comany.com

For multiple addresses:

"user name" <mailid@company.com> , "user 2" <mailid@company.com>

<mailid@company.com> , "user 2" <mailid@company.com>

"user name" <mailid@company.com> , mailid@company.com, <mailid@company.com>

Declaration Files

C declarations

Visual Basic Declarations

Delphi Declarations

PowerBuilder Declarations

C declarations

The 16bits C function declarations for SMTPOP16 are available in SMTPOP.H

The 32bits C function declarations for SMTPOP32 are available in SMTPOP.H

Visual Basic Declarations

The 16bits Visual Basic function declarations for SMTPOP16 are available in SMTP.BAS.

The 32bits Visual Basic function declarations for SMTPOP32 are available in SMTP.BAS.

Delphi Declarations

The 16bits Delphi function declarations for SMTPOP16 are available in SMTPOP16.PAS.

The 32bits Delphi function declarations for SMTPOP32 are available in SMTPOP32.PAS.

PowerBuilder Declarations

The 16bits PowerBuilder function declarations for SMTPOP16 are available in SMTPOP16.SRU.

The 32bits PowerBuilder function declarations for SMTPOP32 are available in SMTPOP32.SRU.

Error Codes

The format for SMTPOP16.DLL/SMTPOP32.DLL error is "XABCC description", where

X: Severity level:

S - System

E - Error

W - Warning

I - Information

A: Error type:

0 - system

1 - syntax

2 - info

B: Internal index:

0 - 9

CC: Error number

00 - 99

Related Topics:

S2001

S2002

S2003

E2000

E2100

E2101

E2102

E2103

E2104

E2106

E1200

E1201

E1202

E0300

E2301

E2303

E2305

E2306

E2307

E2308
E2401
E2402
E2403
E2404
E2405
E2406
E2407
E2410
E2411
E2412
E2413
E2414
I2105
I2304
I2409
W0302

S2001

S2001 internal error: unknown type *[number].*

This is an internal error. Please report this error to H&S Technology, Inc.

S2002

S2002 *[sting]*: unknown mode

This is an internal error. Please report this error to H&S Technology, Inc.

S2003

S2003 Out of memory

Ran out of memory, please increase your RAM or limit the number of tasks running at same time.

E2000

E2000 There is no open handle

E2100

E2100 *[string]*

E2101

E2101 Error opening Data Channel. *[string]*

E2102

E2102 Mail Data failed. *[string]*

E2103

E2103 Command out of Sync.

If you are retrieving mail, check your connection to your POP3 server. It is likely that you do not have a valid connection.

E2104

E2104 Error establishing session with mail server.

E2106

E2106 Invalid address: *[string]*

The mail address is invalid.

E1200

E1200 Invalid handle

The handle is invalid, please make sure you have called one of the open functions to create a valid handle.

E1201

E1201 Syntax: pop3_GetMailSize requires mail message number to be greater than zero

Please refer to pop3_GetMailSize function prototype.

E1202

E1202 Error opening file: *[filename]*

E0300

E0300 System Error.

System error.

E2301

E2301 Connection Severed by server

The remote SMTP server has terminated your session.

E2303

E2303 Error sending "*[string]*", possible connection severed

E2305

E2305 No control connection established

E2306

E2306 Error on socket: "*[string]*"

E2307

E2307 Error opening file "*[string]*": "*[string]*"

E2308

E2308 Error reading reply. Reply buffer is empty."

E2401

E2401 Error creating a socket: *[number]*

E2402

E2402 Error connecting to *[string]*: *[number]*

E2403

E2403 Error invalid socket(*[string]*) connected

E2404

E2404 Error getsockname: *[number]*

E2405

E2405 Error communicating with SMTP server

E2406

E2406 Error setsockopt (re-use address)

E2407

E2407 Error listening: *[number]*

E2410

E2410 Winsock *[number]*, need Winsock DLL version 1.1 and above to run

E2411

E2411 Error setsockopt: *[number]*

TCP/IP option error. Your TCP/IP stack probably does not support this option.

E2412

E2412 Error accept: *[number]*

TCP/IP error. You may need to reboot your system.

E2413

E2413 Error binding: *[number]*

TCP/IP error. You may need to reboot your system.

E2414

E2414 Error unknown host: *[string]*

TCP/IP error. Cannot resolve your TCP/IP host name. Please make sure the host name is correct.

I2105

I2105 *[string]*

I2304

I2304 Session closed successfully

I2409

I2409 Connection closed

W0302

W0302 Internal buffer size is not large enough, reply has been truncated

Glossary of Terms

WINSOCK.DLL

MIME

RFC

DLL

POP3

SMTP

WINSOCK.DLL

Windows Sockets Dynamic Link Library

MIME

Multipurpose Internet Mail Extensions

RFC

Request For Comment

DLL

Dynamic Link Library

POP3

Post Office Protocol version 3

SMTP

Simple Mail Transfer Protocol