



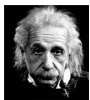
[articles](#) [quick answers](#) [discussions](#) [features](#)
[community](#) [help](#)

Articles / Programming Languages / C#

[Watch](#)


[C++](#) [C#](#) [.NET](#) [SSL](#) [IMAP](#) [MTP](#) [POP3](#) [TLS](#)

vmime.NET - Smtplib, Pop3, Imap Library (for C++ and .NET)



Elmue

Rate me:  4.77/5 (57 votes)

20 Aug 2014 [GPL3](#) 18 min read  239.2K  1.7K  148  85

With this project, C++ and .NET Windows programmers get a very versatile library to send and download emails via SMTP, POP3 and IMAP with TLS and SSL support.

Downloads

Vmime.NET_All.zip (9 MB): The entire project with all files required to compile for 32 Bit platform.

Vmime.NET_64.zip (6 MB): Only a few additional files required to compile for 64 Bit platform.

Summary

With this project, **C++** and **.NET** Windows programmers get a very versatile library to send and download emails via **SMTP**, **POP3** and **IMAP** with **TLS** and **SSL** support.

Features

- This is a **high quality** library that can be used in production.
- **SMTP**, **POP3**, **IMAP** protocols implemented.

- **TLS** and **SSL** encryption (using openssl library) with 185 built-in **X.509 root certificates**.
- **SASL** authentication (Simple Authentication and Security Layer).
- **Email** builder builds RFC-2822 and multipart messages.
- **HTML** email generation with embedded objects.
- Full support for **attachments** with automatic mime-type detection of 1010 built-in **mime types**.
- **Email** parser allows to extract attachments or text.
- You will learn in 15 minutes how to use the library. It follows the **KISS principle** ("Keep it simple, stupid").
- This project is mainly written in **pure C++** and has a **mangled C++** wrapper that exposes the functionality to .NET (**C#** and **Visual Basic .NET**).
- Fully **RFC**-compliant implementation.
- **8-bit MIME**, encoded word extensions, pipelining and chunked message transfer.
- **Trace** output showing the entire communication with the server.
- Full **Unicode** support for CJK (Chinese, Japanese, Korean).
- Very **clean code** written by very experienced programmers.
- Demo application in C++.
- Demo application in C#.
- All required dependencies (e.g. openssl) are included in the download. You don't have to download other sources to compile the project.
- The download contains **vmime.NET_2.dll** for .NET **framework 2.0**, 3.0, 3.5.
- The download contains **vmime.NET_4.dll** for .NET **framework 4.0**, 4.5, 4.6., etc..
- The second download contains the **64 Bit** version of each of them.
- The DLL is **strong named** (signed).
- GNU **GPL** license. (Commercial licenses available)
- Update february 2016: Now the project compiles without the deprecated compiler switch '/clr:oldsyntax'.

Why using vmime ?

I was searching for a SMTP/POP3/IMAP library. What I found were either very expensive **commercial** libraries or free libraries without **encryption** support. Today many SMTP servers demand a TLS or SSL connection. With an email library that does not support encryption you will not be able to send an email to a Gmail SMTP server.

Finally I found vmime, which is a multi platform library for SMTP, POP3, IMAP, SendMail and MailDir with very cleanly written code and hosted on Github, mainly written by Vincent Richard, who did a great work.

A nightmare begins...

But this library is a Gnu project that was developed and tested mainly in the **Mac / Linux** world. Theoretically it should be possible to compile it on Windows with the **Cmake** compiler, but that threw a lot of cryptic error messages that I never saw before, so I gave up.

Additionally CMake is useless if you want to write a Managed C++ project for .NET. So I had to find a way to compile this stuff on **Visual Studio**. If you ever ported code from Linux to Windows you know what nightmare this is.

After eliminating all the errors in Visual Studio that you see whenever you compile a Linux project on Windows, I had the problem that the **Config.hpp** file was missing which contains all the project settings. I had to find out how to construct this file manually.

Then vmime depends on several other Gnu projects which I had to download separately:

1. **GnuTLS** or **OpenSSL** library for encryption,
2. **iconv**, a character set converter library,
3. **gsasl**, an authentication library

All these libraries exist precompiled for Windows as DLLs and come with a Lib file for the linker. Some of them come with library files with names like "**libgsasl.a**". I never saw this file extension before and thought: These files are made for Windows so I change that to "libgsasl.lib". And the worst of all is that Visual Studio eats these files without complaining. But the compiled binary file will not run.

Finally I found out that these *.a files are for a Gnu compiler on Windows.

And where do I find the Lib files for Visual Studio?

They simply do not exist.

After investigating I found that I can create the Lib file from the **Def** file with a Visual Studio command line tool. See "libvmime\src\gsasl**CreateLIB.bat**"

I started my first tests with **GnuTLS** which was an error, because:

1. GnuTLS depends on 4 more libraries (total 5 DLL's!)
2. None of these DLLs exists as 64 Bit version
3. It does not run correctly on Windows (throws Error "The TLS connection was non properly terminated" in the middle of an email transfer)

I wasted a lot of time with this library, finally I moved to **OpenSSL** which works perfectly.

The **iconv** library also caused me problems because there is no precompiled 64 Bit version for Windows and this library has apparently never been compiled for 64 Bit. Finally I replaced it with my own code that uses the Windows Codepage conversion API instead.

Problems with vmime

Then I started struggling with vmime code itself:

A severe lack was that vmime had no **Unicode** support. All strings are std::string while std::wstring was used in no place. Filenames were translated with the current ANSI codepage and passed to the ANSI file API (e.g. CreateFileA).

To permit that vmime can also be used by **chinese** or **japanese** users I had to find a way to pass unicode into vmime. I ended up converting all input strings in a wrapper class into UTF-8. Then, in

the windows platform handler I convert the UTF-8 strings back to Unicode and pass them to widechar API (CreateFileW).

Another severe lack of vmime was the missing **Trace** output. When an error occurred with the server you had no idea what was the reason because you did not see the communication with the server. I implemented Trace support as you see in the screenshots below.

Another lack was that vmime code was not **abortable**. If the server did not respond the user had to wait until the timeout elapsed - up to 30 seconds! I fixed that.

Another problem was that in vmime **error handling** was not properly implemented.

Then I found several **bugs**, like for example openssl throwing "SSL3_WRITE_PENDING:bad write retry". I contacted Vincent - the author of vmime - who gave me excellent support with all problems and fixed some of these bugs.

Then I added new features that are indispensable for an easy usage like automatic **mime type detection** by file extension and loading **root certificates** from the resources.

The vmime library has been designed to be very flexible and expandable. This is good but has the disadvantage that the usage becomes **awkward**. For a simple task like extracting the plain text of an email you have to write a loop, enumerate all message parts, use a MessageParser, use a OutputStreamStringAdapter, a ContentHandler, a SmartPointer class, dynamic casts and character set conversion. All this is clumsy and an experienced programmer - new to vmime - easily needs some hours to find out how to do this. And for C++ beginners it will be nearly impossible to use vmime.

So I added **wrapper classes** that hide all this complicated stuff in a single function. ([KISS principle](#))

Then I discovered a VERY weird phenomenon: Whenever in vmime an exception was thrown, I had **memory leaks** and TCP sockets that were not closed because several destructors were never called (no stack unwinding). You can read on [Stackoverflow](#) how I solved this.

I did a lot more things but to mention all this would be too much here.....

All my changes in the source code are marked with a comment: **// FIX by Elmue**, if you search so you find more than 240 modifications.

Finally...

...finally I was working two months on this project. I did really frustrating work and I nearly gave up more than once.

Now you are in the fortunate situation, that you can download a ready-to-use library that works perfectly and is **very easy to use** and intuitive.

EmailBuilder

With very few lines of code you can create a mime email with a plain text part, a Html part, embedded Html objects (e.g. images) and attachments. Both, plain text and Html text are optional. Modern email clients do not display the plain text part if a Html part is present.

C#

C#



```
using vmimeNET;
using (EmailBuilder i_Email = new EmailBuilder("John Miller <jmiller@gmail.com>",
                                              "vmime.NET Test Email"))
{
    i_Email.AddTo("recipient1@gmail.com"); // or "Name <email>"
    i_Email.AddTo("recipient2@gmail.com");

    i_Email.SetPlainText("This is the plain message part.\r\n" +
                        "This is Chinese: \x65B9\x8A00\x5730\x9EDE");

    i_Email.SetHtmlText ("This is the <b>HTML</b> message part.<br/>" +
                        "<img src=\"cid:VmimeLogo\"/><br/>" +
                        "(This image is an embedded object)<br/>" +
                        "This is Chinese: \x65B9\x8A00\x5730\x9EDE");

    i_Email.AddEmbeddedObject("E:\\Images\\Logo.png", "", "VmimeLogo");

    i_Email.AddAttachment("E:\\Documents\\ReadMe.txt", "", "");

    i_Email.SetHeaderField(Email.eHeaderField.Organization, "ElmueSoft");

    String s_Email = i_Email.Generate();
} // i_Email.Dispose()
```

C++

C++



```
using namespace vmime::wrapper;
cEmailBuilder i_Email(L"John Miller <jmiller@gmail.com>", L"vmime.NET Test Email",
                    IDR_MIME_TYPES);

i_Email.AddTo(L"recipient1@gmail.com"); // or "Name <email>"
i_Email.AddTo(L"recipient2@gmail.com");

i_Email.SetPlainText(L"This is the plain message part.\r\n"
                    L"This is Chinese: \x65B9\x8A00\x5730\x9EDE");

i_Email.SetHtmlText (L"This is the <b>HTML</b> message part.<br/>"
                    L"<img src=\"cid:VmimeLogo\"/><br/>"
                    L"(This image is an embedded object)<br/>"
                    L"This is Chinese: \x65B9\x8A00\x5730\x9EDE");

i_Email.AddEmbeddedObject(L"E:\\Images\\Logo.png", L"", L"VmimeLogo");

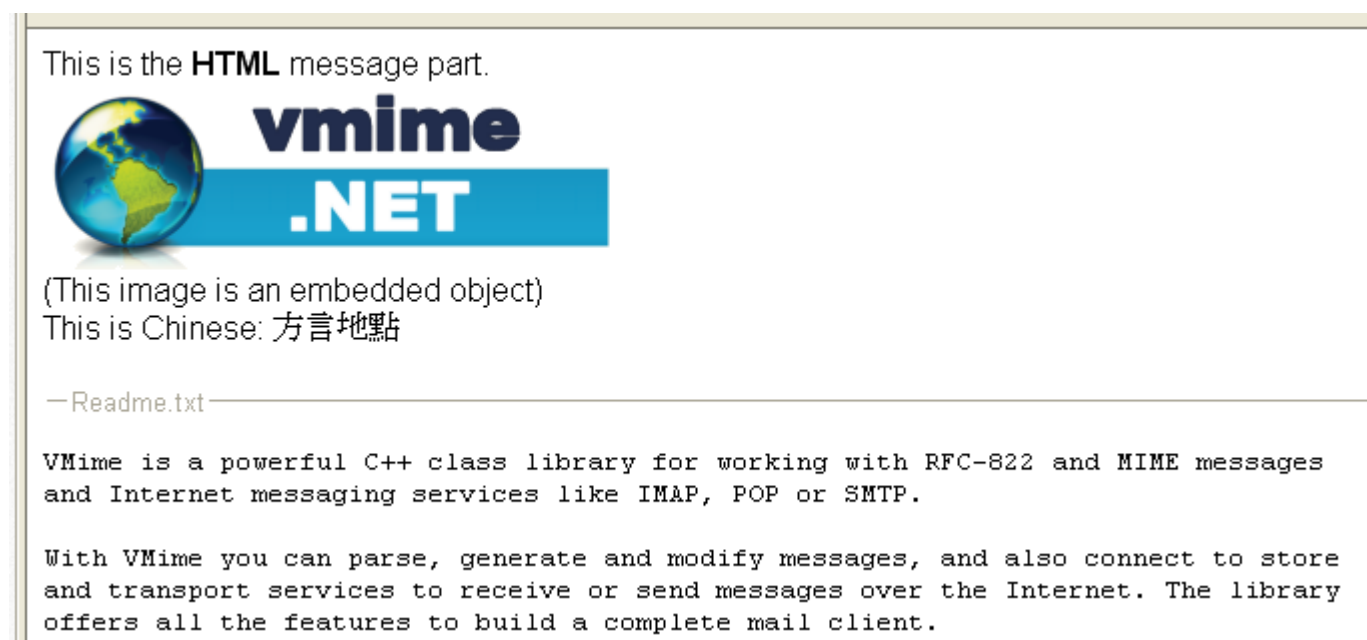
i_Email.AddAttachment(L"E:\\Documents\\ReadMe.txt", L"", L "");

i_Email.SetHeaderField(cEmail::Head_Organization, L"ElmueSoft");

std::wstring s_Email = i_Email.Generate();
```

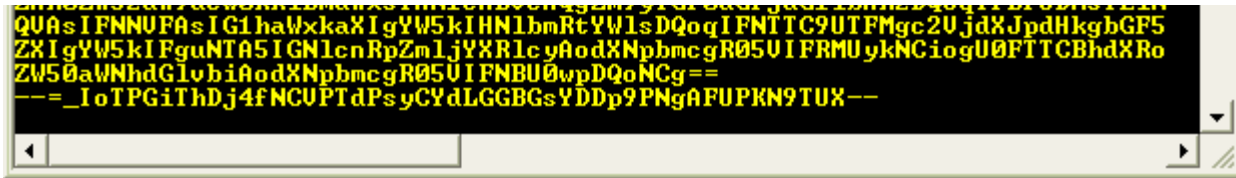
In this example "VmimeLogo" is the **content identifier** of the embedded image. In Html you can reference embedded images as: .

The above code will generate an email that looks like this in Thunderbird:



Here the source code in the variable s_Email printed to the console:

[illegible]



SMTP

Sending our email is done with 3 lines of code: This could not be more simple. ([KISS principle](#))

C#

C# 

```
using (Smtp i_Smtp = new Smtp(s_Server, u16_Port, e_Security, b_AllowInvalidCertificate))
{
    i_Smtp.SetAuthData(s_User, s_Password);
    i_Smtp.Send(i_Email);
} // i_Smtp.Dispose()
```

C++

C++ 

```
cSmtp i_Smtp(u16_Server, u16_Port, e_Security, b_AllowInvalidCertificate, IDR_ROOT_CA);
i_Smtp.SetAuthData(u16_User, u16_Password);
i_Smtp.Send(&i_Email);
```

If `u16_Port == 0` the default port is used.

If the server sends an invalid certificate:

- if `b_AllowInvalidCertificate == true` an error is written to the Trace output,
- if `b_AllowInvalidCertificate == false` an exception is thrown and the email is not sent.


```

e:\Workspace\Vmime.NET\output32\DemoCpp.exe
SMTP Connecting to 'smtp.googlemail.com' on port 25 (not using SSL)
SMTP read < "220 mx.google.com ESMTP disn10090383qai.7 - gsntp"
SMTP send > "EHLO Starwalker"
SMTP read < "250-mx.google.com at your service, [200.112.60.101]"
SMTP read < "250-SIZE 35882577"
SMTP read < "250-8BITMIME"
SMTP read < "250-STARTTLS"
SMTP read < "250-ENHANCEDSTATUSCODES"
SMTP read < "250 CHUNKING"
SMTP send > "STARTTLS"
SMTP read < "220 2.0.0 Ready to start TLS"
CERT Server X.509 certificate at chain pos 1: CN=Google Internet Authority G2,O=Google Inc,C=US
CERT Server X.509 certificate at chain pos 2: CN=GeoTrust Global CA,O=GeoTrust Inc.,C=US
CERT Server X.509 certificate at chain pos 3: OU=Equifax Secure Certificate Authority,O=Equifax,C=US
CERT 185 Root Certificates (Certificate Authorities) loaded.
CERT Server certificate is signed with root certificate: OU=Equifax Secure Certificate Authority,O=Equifax,C=US
SMTP Encrypted TLS session started.
SMTP send > "EHLO Starwalker"
SMTP read < "250-mx.google.com at your service, [200.112.60.101]"
SMTP read < "250-SIZE 35882577"
SMTP read < "250-8BITMIME"
SMTP read < "250-AUTH LOGIN PLAIN XOAUTH XOAUTH2 PLAIN-CLIENTTOKEN"
SMTP read < "250-ENHANCEDSTATUSCODES"
SMTP read < "250 CHUNKING"
SASL All implemented mechanisms: ANONYMOUS, EXTERNAL, LOGIN, PLAIN, SECURID, DIGEST-MD5, CRAM-MD5, SCRAM-SHA-1
SASL Available mechanisms on this server: LOGIN, PLAIN
SMTP send > "AUTH PLAIN"
SMTP read < "334 "
SMTP send > <Authentication Data> <31 Bytes>
SMTP read < "235 2.7.0 Accepted"
SMTP send > "MAIL FROM:<sender@gmail.com> SIZE=24995"
SMTP read < "250 2.1.0 OK disn10090383qai.7 - gsntp"
SMTP send > "RCPT TO:<recipient@gmail.com>"
SMTP read < "250 2.1.5 OK disn10090383qai.7 - gsntp"
SMTP send > <Chunked Message Data> <24995 Bytes>
SMTP Message transfer starting.
SMTP send > "BDAT 25136 LAST"
SMTP read < "250 2.0.0 OK disn10090383qai.7 - gsntp"
SMTP Message transfer finished. (25136 Bytes sent)
SMTP send > "QUIT"
SMTP read < "221 2.0.0 closing connection disn10090383qai.7 - gsntp"
Press Enter!

```

Trace

The Trace output you see above is returned by vmime.NET via a callback. You can do whatever you like with it: write it to the console or display it in a log window or save it to a **logfile**. Or you turn off the compiler switch **VMIME_TRACE** and Trace is completely disabled.

A note about mass emails

If you plan to send for example a newsletter with 500 recipients you will encounter lots of problems:

- All Freemail providers (Gmail, Yahoo, Gmx, etc) limit to a maximum amount of recipients per email (e.g. 100)
- All Freemail providers limit to a maximum amount of emails per day.
- Freemail accounts may be closed when you break the rules.

These measures are indispensable to fight **spam** that is mainly sent by Trojans and Botnets via SMTP. (Read about [Trojans and Botnets](#) in my article)

So the best solution to send a newsletter is to install your own email server. (I recommend [MDaemon Email Server](#))

But even with this solution you must be carefull because when your IP address gets listed on a blacklist your server may get blocked for multiple days.

Security

You have 4 options to send the email to the SMTP server: (`enum e_Security`)

1. on normal port (default 25) **unencrypted**. This deprecated option should be used only if the server is in the same intranet as you and only if there is no way to use the other options.
2. on secure port (default 465) using **SSL** encryption.
3. on normal port (default 25) using **TLS** encryption if possible.
4. on submission port (default 587) requires **TLS** encryption otherwise fails.

Both, SSL (**S**ecure **S**ockets **L**ayer) and TLS (**T**ransport **L**ayer **S**ecurity) send the authentication (user/password) and the email message encrypted.

SSL starts immediately an encrypted connection.

TLS is the successor of SSL. It starts with an unencrypted connection, sends the SMTP command "STARTTLS" and then encrypts the traffic on the same port.

Read Wikipedia about [SSL](#) and [TLS](#).

Server Certificates

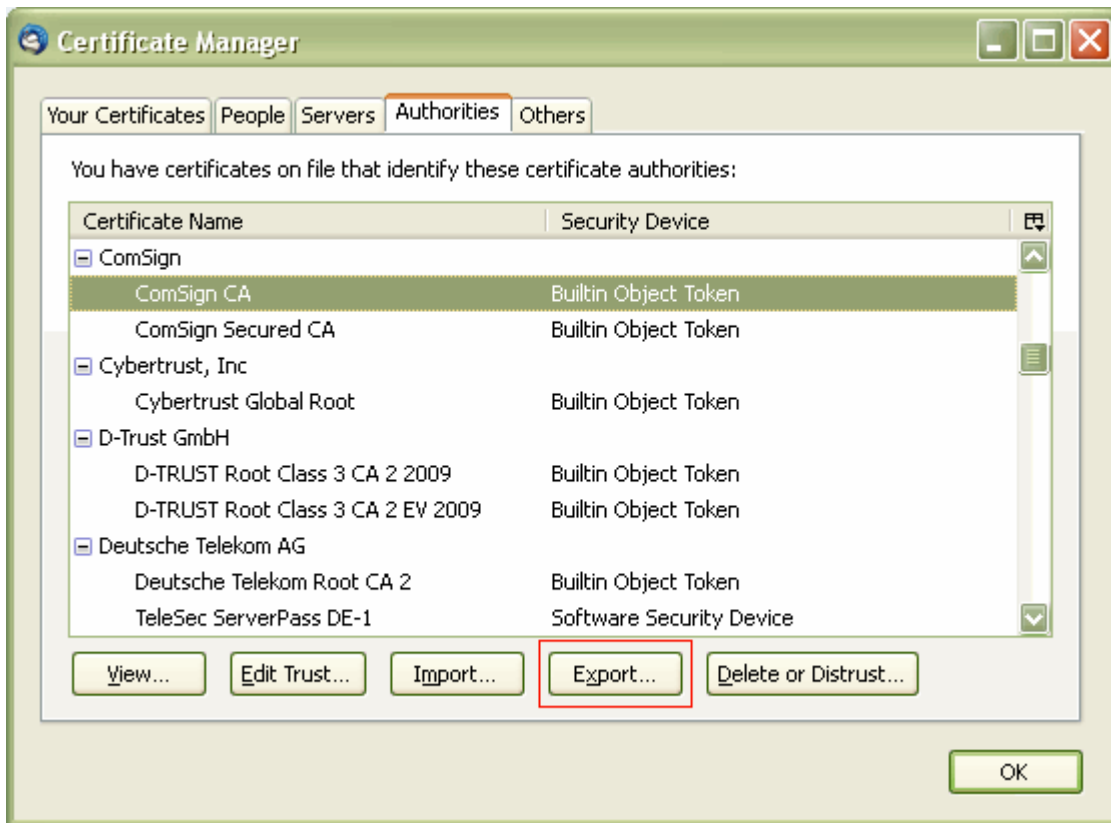
When using SSL or TLS the server authenticates itself with a **X.509** certificate. vmime.NET has **185 built-in root certificates** and checks that the server certificate is digitally signed with one of these root authorities (e.g. Verisign), that the expiration **date** of the certificate has not expired and that the **hostname** of the server is the same as in the certificate. With this certificate the client can prove that there is no [man-in-the-middle attack](#).

Read Wikipedia about [X.509](#)

IMPORTANT:

If you download this project several months after I published it (january 2014) you should **update the certificates** because they may expire, be revoked or new root authorities may exist:

1. Delete all files in the folder Converter\RootCertificates\
2. Install the latest version of Mozilla **Thunderbird**
3. Go to Tools -> Options -> Advanced -> View Certificates
4. Click every certificate and export it as **PEM** file into the folder Converter\RootCertificates\
(If a certificate file with the same name already exists, give it another name)
5. Run the Converter tool in the folder Converter\Release\ that reads these certificate files and writes them into a single text file.
6. Compile the project anew which embeds this text file into the resources of the binary (Dll or Exe)
7. Make sure that Visual Studio shows "**Compiling Resources...**" in the Output pane!



Pop3

This code enumerates all emails in the POP3 Inbox:

C#



```
using (Pop3 i_Pop3 = new Pop3(s_Server, u16_Port, e_Security, b_AllowInvalidCertificate))
{
    i_Pop3.SetAuthData(s_User, s_Password);

    int s32_EmailCount = i_Pop3.GetEmailCount();
    for (int M=0; M<s32_EmailCount; M++)
    {
        using (EmailParser i_Email = i_Pop3.FetchEmailAt(M))
        {
            // do something with the email...
        } // i_Email.Dispose()
    }
} // i_Pop3.Dispose()
```

C++



```
cPop3 i_Pop3(u16_Server, u16_Port, e_Security, b_AllowInvalidCertificate, IDR_ROOT_CA);
i_Pop3.SetAuthData(u16_User, u16_Password);

int s32_EmailCount = i_Pop3.GetEmailCount();
for (int M=0; M<s32_EmailCount; M++)
```

```
{
    GuardPtr<cEmailParser> i_Email = i_Pop3.FetchEmailAt(M);
    // do something with the email...
}
i_Pop3.Close(); // Close connection to server
```

NOTE:

Some POP3 servers do not respond on port 110 (e.g. Gmail) or reject a connection. The only possible security mode for these servers is **SSL** (port 995).

IMPORTANT:

Gmail has a **very buggy** POP3 behaviour: All emails can be downloaded only **ONCE** via POP3, no matter what setting you chose in your POP3/IMAP configuration! The emails are there in the web interface but pop.googlemail.com tells you that you have no emails in your Inbox. Or it may happen the opposite that POP3 shows you emails that have been deleted years ago (but only once)! Gmail's IMAP behaves normal.

EmailParser

The class **EmailParser** allows to extract any part of an email.

C#

C#

Shrink ▲ 

```
// ===== HEADER (POP3 command TOP) =====

String s_Subject = i_Email.GetSubject(); // "vmime.NET Test Email"

// s_From[0] = "jmillergmail.com",
// s_From[1] = "John Miller"
String[] s_From = i_Email.GetFrom();

int s32_Timezone; // deviation (+/-) from GMT in minutes
DateTime i_Date = i_Email.GetDate(out s32_Timezone);

String s_UserAgent = i_Email.GetUserAgent(); // "vmime.NET 0.9.2 (www.codeproject.com)"

List<String> i_Emails = new List<String>(); // "jmillergmail.com",
"ccastillogmail.com", ...
List<String> i_Names = new List<String>(); // "John Miller", "Cristina Castillo",
...
i_Email.GetTo(i_Emails, i_Names); // get all TO: recipients

// ===== Size (POP3 command LIST) =====

UInt32 u32_Size = i_Email.GetSize(); // Email size in bytes

// ===== Uid (POP3 command UIDL) =====

String s_UID = i_Email.GetUID(); // Unique identifier of this email on the server

// ===== Body (POP3 command RETR) =====

String s_PlainText = i_Email.GetPlainText(); // "This is the plain message part..."
```

```
String s_HtmlText = i_Email.GetHtmlText(); // "This is the <b>HTML</b> message part..."

UInt32 u32_ObjCount = i_Email.GetEmbeddedObjectCount(); // number of embedded objects in
the HTML part

String s_ObjId; // "VmimeLogo"
String s_ObjType; // "image/png"
Byte[] u8_ObjData;
UInt32 u32_Object = 0;
while (i_Email.GetEmbeddedObjectAt(u32_Object++, out s_ObjId, out s_ObjType, out
u8_ObjData))
{
    // do something with the embedded object
}

UInt32 u32_AttCount = i_Email.GetAttachmentCount(); // number of attachments

String s_AttName; // "Readme.txt"
String s_AttType; // "text/plain"
Byte[] u8_AttData;
UInt32 u32_Attach = 0;
while (i_Email.GetAttachmentAt(u32_Attach++, out s_AttName, out s_AttType, out
u8_AttData))
{
    // do something with the attachment
}
```

C++

See DemoCpp project.

NOTE:

The console uses codepage 1252 and cannot display chinese characters.

Performance

When you fetch an email with `i_Pop3.FetchEmailAt(M)` only the mail **header** is downloaded (POP3 command "**TOP**"). This is quite fast and allows to access the Subject, From, To, Cc, Date, UserAgent, and more header fields.

When you call `i_Email.GetSize()` the POP3 command "**LIST**" is sent to the server to obtain the size of the email.

When you call `i_Email.GetUID()` the POP3 command "**UIDL**" is sent to the server to obtain the unique identifier of the email.

And when you access the body part of the email, the POP3 command "**RETR**" retrieves the entire email which may be slow if the email has several megabytes. The body contains the plain text, HTML text, embedded objects and attachments.

So, to improve **performance** you should access the body part only if you really need it.

IMAP

Imap has the same commands as Pop3.
Additionally there are:

C#

C#



```
// Enumerate all folders on the server
String[] s_Folders = i_Imap.EnumFolders();

// Select the folder to retrieve emails from
i_Imap.SelectFolder("[Gmail]/Sent Mail");

// Get the current folder
String s_CurFolder = i_Imap.GetCurrentFolder();
```

C++

C++



```
// Enumerate all folders on the server
void i_Imap.EnumFolders(vector<wstring>& i_FolderList);

// Select the folder to retrieve emails from
void i_Imap.SelectFolder(L"[Gmail]/Sent Mail");

// Get the current folder
wstring s_CurFolder = i_Imap.GetCurrentFolder();
```


As you see IMAP is far more complex than POP3. A new connection is established to the server each time a new folder is opened. (**IMAP (1)** is the first, **IMAP (2)** the second connection, etc...)

Performance

Here applies the same as for POP3, except that for obtaining Size and UID no additional command has to be sent to the server.

Deleting emails

The following code shows how to delete all emails with the subject "**vmime.NET Test Email**" that you have sent by the SMTP demo before to yourself.

C#

C#



```
// i_Service = POP3 or IMAP
int s32_EmailCount = i_Service.GetEmailCount();

// Run the Loop reverse to avoid M indexing an already deleted email
for (int M=s32_EmailCount-1; M>=0; M--)
{
    using (EmailParser i_Email = i_Service.FetchEmailAt(M))
    {
        if (i_Email.GetSubject() == "vmime.NET Test Email")
            i_Email.Delete(); // POP3: mark for deletion, IMAP: Delete now
    } // i_Email.Dispose()
}

i_Service.Close(); // POP3: Expunge all emails marked for deletion
```

C++

```
// i_Service = POP3 or IMAP
int s32_EmailCount = i_Service.GetEmailCount();

// Run the Loop reverse to avoid M indexing an already deleted email
for (int M=s32_EmailCount-1; M>=0; M--)
{
    GuardPtr<cEmailParser> i_Email = i_Service.FetchEmailAt(M);

    if (i_Email->GetSubject() == L"vmime.NET Test Email")
        i_Email->Delete(); // POP3: mark for deletion, IMAP: Delete now
}

i_Service.Close(); // POP3: Expunge all emails marked for deletion
```

IMPORTANT:

POP3: When you call `i_Pop3.Close()` all emails marked for deletion are expunged definitely from the server. If you do not call `Close()` they will not be deleted!

IMAP: When you call `i_Email.Delete()` the email is deleted immediately.

Download URL's

Here are the URL's where I downloaded the several components that are required to build the project.

You don't have to download them. This is just for your information.

vmime

<https://github.com/kisli/vmime>

I checked out version 0.9.2 with the GitHub tool

openssl

Downloadpage: <http://www.openssl.org/source> (Source)

Downloadpage: <http://slproweb.com/products/Win32OpenSSL.html> (Binaries)

Sourcecode: <http://www.openssl.org/source/openssl-1.0.1e.tar.gz>

Win32Binaries: http://slproweb.com/download/Win32OpenSSL-1_0_1e.exe

Win64Binaries: http://slproweb.com/download/Win64OpenSSL-1_0_1e.exe

libgsasl

Downloadpage: <http://www.gnu.org/software/gsasL/#downloading>

Sourcecode: <ftp://ftp.gnu.org/gnu/gsasL/gsasL-1.6.0.tar.gz>

Win32Binaries: <ftp://ftp.gnu.org/gnu/gsasL/gsasL-1.6.0-x86.zip>

Win64Binaries: <ftp://ftp.gnu.org/gnu/gsasL/gsasL-1.6.0-x64.zip>

ATTENTION:

The binaries in gsasl-1.6.1-x64.zip and gsasl-1.8.0-x64.zip on the ftp server are wrongly compiled (they are 32 Bit, not 64 Bit!)

Mime types

<http://svn.apache.org/repos/asf/tomcat/trunk/conf/web.xml>

Visual Studio 2005

If you use Visual Studio 2005 it is indispensable for this project to have **VS Service Pack 1** installed. Otherwise you will see Intellisense hanging forever. This bug has been fixed in SP1. See [MSDN](#).

Compiling and Dependencies

vmime.NET can be compiled as x86 (Win32) or x64 (Win64) and as Debug or Release. There are two dependencies: GNU Sasl and OpenSSL.

The file cCommon.cpp contains the #pragma comment commands that tell the linker which Lib files to include depending on the current project settings.

The **GSASL** library consists of **libgsasl-7.dll** and the LIB files libgsasl-7_32.lib and libgsasl-7_64.lib. If you should replace the DLL with another version you must also replace the Def file and run the script CreateLIB.bat to create the Lib files anew. Additionally the header files may have changed.

The **OpenSSL** library may be compiled dynamic (using **ssleay32.dll** and **libeay32.dll**) or static (**no DLL's** required)

When you run the OpenSSL installer for Windows you get a lot of LIB files:

Dynamic for compiler switch /MD:

OpenSSL-Win32\lib\VC\libeay32MD.lib (800 kB) // Release

OpenSSL-Win32\lib\VC\libeay32MDd.lib (800 kB) // Debug

OpenSSL-Win32\lib\VC\ssleay32MD.lib (70 kB) // Release

OpenSSL-Win32\lib\VC\ssleay32MDd.lib (70 kB) // Debug

Static for compiler switch /MD:

OpenSSL-Win32\lib\VC\static\libeay32MD.lib (20 MB) // Release

OpenSSL-Win32\lib\VC\static\libeay32MDd.lib (20 MB) // Debug

OpenSSL-Win32\lib\VC\static\ssleay32MD.lib (4 MB) // Release

OpenSSL-Win32\lib\VC\static\ssleay32MDd.lib (4 MB) // Debug

The same amount of LIB files exist additionally for compiler switch **/MT** but this switch cannot be used in Managed C++ projects (vmime.NET.dll).

And the same amount of Lib files exists additionally for **64 Bit**.

So there are totally 32 LIB files.

IMPORTANT:

If you should replace openssl with another version you must verify if the header files have changed.

Compiling vmime.NET.dll

I **strongly recommend** to link vmime.NET.dll **statically** because

1. The installer of your software has to deliver 2 DLLs less.
2. The openssl DLLs themselves are dependent of the VC++ runtime 2008 (Msvcr90.dll).

If you compile on **Visual Studio 2008** this does not matter because openssl is also compiled on VS2008, but on any other Visual Studio version it does:

vmime code itself also depends on the VC++ runtime but the version of the C++ runtime will depend on your Visual Studio version.

For example VS 2005 creates a dependency to Msvcp80.dll, Msvcr80.dll and Msvcm80.dll.

So, what would happen if you compile on VS 2005 and link dynamically to ssleay32.dll and libeay32.dll ?

Then your software would depend on Msvcp80.dll, Msvcr80.dll and Msvcm80.dll due to vmime and **additionally** on Msvcr90.dll which is required by ssleay32.dll and libeay32.dll.

This would mean that the users of your software have to install **two** VC++ runtimes!

Compiling Pure C++ Projects

All the above applies to vmime.NET.dll where the compiler switch /MT cannot be used (Visual Studio restriction).

But if you use vmime in a pure C++ project you can compile with /MT and use the openssl libraries **libeay32MT.lib** and **ssleay32MT.lib** instead.

This links the C++ runtime statically into your application and you do NOT need any of the MsvcXX.dll's.

DependencyWalker

If you are not sure on which DLLs your compiled binaries depend use [DependencyWalker](#) to check that:

This screenshot shows the dependencies if compiled with openssl static LIB files on VS2005 as Release, 32 Bit.

Dependencies

Project	Required DLL's	C++ Redistributable
Pure C++ project with /MT, openssl static LIB files xxxMT.lib	libgsasl-7.dll	none
Pure C++ project with /MD, openssl static LIB files xxxMD.lib	libgsasl-7.dll	for your VS version
Pure C++ project with /MT, openssl dynamic LIB files xxxMT.lib	libgsasl-7.dll, ssleay32.dll, libeay32.dll	2008
Pure C++ project with /MD, openssl dynamic LIB files xxxMD.lib	libgsasl-7.dll, ssleay32.dll, libeay32.dll	2008 and for your VS version
<hr/>		
vmime.NET Project with /MD, openssl static LIB files xxxMD.lib	libgsasl-7.dll	for your VS version
vmime.NET Project with /MD, openssl dynamic LIB files xxxMD.lib	libgsasl-7.dll, ssleay32.dll, libeay32.dll	2008 and for your VS version

The files in bold are included in the download on Codeproject.

VC++ runtime installer

Compiler	installed DLLs	Download
VS 2005 - 32 Bit	MsVcr80.DLL, MsVcp80.DLL, MsVcm80.DLL	Download
VS 2005 - 64 Bit		Download
VS 2008 - 32 Bit	MsVcr90.DLL, MsVcp90.DLL, MsVcm90.DLL	Download
VS 2008 - 64 Bit		Download
VS 2010 - 32 Bit	MsVcr100.DLL, MsVcp100.DLL, MsVcm100.DLL	Download

VS 2010 - 64 Bit	MsVcr110.DLL, MsVcp110.DLL, MsVcm110.DLL	Download
VS 2012 - 32 Bit		Download
VS 2012 - 64 Bit		

IMPORTANT:

Microsoft's VC++ runtime installer does NOT install the Debug versions of these DLL's. You MUST compile as **Release** before delivering your software!

ATTENTION:

If these DLL's are not installed on the target machine the user will NOT get an intelligent error message telling him what is wrong. Instead Microsoft .NET throws the most stupid exceptions like: **"vmime.NET.dll or one of its dependencies could not be loaded"** or **"System.IO.FileLoadException:....This application has failed to start because the application configuration is incorrect. Reinstalling the application may fix this problem"**. The users of your application will never suspect that the cause is a missing MsVcXX.dll!

There are 2 ways to assure that the MsVcr/MsVcp/MsVcm DLLs exist on the target computer:

1. The users of your application must download and install the correct version of the **Visual C++ Redistributable Package** (see download links in table above). The files will be installed "side by side" into C:\Windows\WinSxS\Cryptic Folder.
2. Or you install the following files directly into the application folder as private assemblies: **Microsoft.VCXX.CRT.manifest, msvcrXX.dll, msvcpXX.dll, msvcmXX.dll**. You find these files in your Visual Studio folder under **VC\Redist**. The versions of these DLLs must be EXACTLY the same as those in the manifest compiled into vmime.NET.dll. You can read the following articles in the MSDN: [Private Assemblies](#) and [Assembly Searching Sequence](#) but probably you will get quite confused.

Try this [tool](#) that checks which VC++ runtimes are installed.

Elmü

License

This article, along with any associated source code and files, is licensed under [The GNU General Public License \(GPLv3\)](#)

Written By

Elmue

Software Developer (Senior) ElmüSoft

 Chile






















Software Engineer since 40 years.

Comments and Discussions

Add a Comment or Question ?			Email Alerts	Search Comments 🔍	
Spacing			Relaxed ▼	Layout	Normal ▼
			Per page	25 ▼	Update
			First Prev Next		
?	How to set charset for GBK for subject 📌	Member 15623749	25-May-22 22:41		
✓	Unicode Encoding of Subject 📌	Elmue	26-May-22 12:45		
📄	Re: Unicode Encoding of Subject 📌	Member 15623749	9-Jun-22 23:28		
?	attachment corrupt when send email with SMTP 📌	Member 15623749	4-May-22 0:38		
✓	Re: attachment corrupt when send email with SMTP 📌	Elmue	5-May-22 14:13		
📄	Re: attachment corrupt when send email with SMTP 📌	Member 15623749	7-May-22 11:50		
✓	Just uploaded new version with bugfix 📌	Elmue	8-May-22 1:02		
📄	Re: Just uploaded new version with bugfix 📌	Member 15623749	12-May-22 3:13		
?	Parsing file 📌	Member 14541384	25-Jul-19 10:42		
✓	Re: Parsing file 📌	Elmue	26-Jul-19 11:36		
?	Error running into windows 7 , 64bit 📌	Member 2281771	30-May-17 19:14		
✓	Re: Error running into windows 7 , 64bit 📌	Elmue	30-May-17 20:21		
📄	Re: Error running into windows 7 , 64bit 📌	Member 2281771	2-Jun-17 10:09		
💡	Compiling as AnyCpu, x86, x64 - Be carefull ! 📌	Elmue	2-Jun-17 14:18		
?	link error with visual studio 15 / win32 📌	Marc Navarro	20-Feb-17 1:55		
✓	Re: link error with visual studio 15 / win32 📌	Elmue	22-Feb-17 9:44		
📄	Re: link error with visual studio 15 / win32 📌	Marc Navarro	27-Feb-17 13:00		
📄	Re: link error with visual studio 15 / win32 📌	Elmue	28-Feb-17 9:05		

24. 2. 2. 오후 1:52

vmime.NET - Smtplib, Pop3, Imap Library (for C++ and .NET) - CodeProject

 Re: link error with visual studio 15 / win32 	 Marc Navarro	5-Mar-17 4:15
 Re: link error with visual studio 15 / win32 	 Elmue	6-Mar-17 15:31
 Support OAUTH2 	 Member 3018048	8-Apr-16 19:25
 Not working for Yahoo mail server 	 Member 2281771	2-Apr-16 16:37
 Re: Not working for Yahoo mail server 	 Elmue	22-Feb-17 9:35
 Corrupted attachment with POP3 	 Member 12328699	15-Feb-16 10:51
 Re: Corrupted attachment with POP3 	 Elmue	15-Feb-16 21:45

Last Visit: 1-Jan-00 0:00 Last Update: 1-Feb-24 23:47

[Refresh](#)

1 2 3 4 Next ▶

 General

 News

 Suggestion

 Question

 Bug

 Answer

 Joke

 Praise

 Rant



Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.

Permalink	Layout: fixed fluid	Article Copyright 2014 by Elmue
Advertise		Everything else Copyright ©
Privacy	Posted 1 Feb 2014	CodeProject , 1999-2024
Cookies		
Terms of Use		Web04 2.8:2024-01-30:1