Subject: HTTPS?
Posted by zsolt on Fri, 01 Dec 2006 12:59:12 GMT
View Forum Message <> Reply to Message

Is it possible in U++, to get data from secure HTTP servers?

Subject: Re: HTTPS?
Posted by mirek on Fri, 01 Dec 2006 13:03:48 GMT
View Forum Message <> Reply to Message

zsolt wrote on Fri, 01 December 2006 07:59Is it possible in U++, to get data from secure HTTP servers?

AFAIK yes, the code is in the Web/SSL, but you need open-ssl library installed. I have never done that, I will contact Tom for details.

Mirek

Subject: Re: HTTPS?
Posted by zsolt on Fri, 01 Dec 2006 13:30:43 GMT
View Forum Message <> Reply to Message

Thanks in advance.
I checked the things and think, the problem is, that HttpClient class does not support SSL, but I'm not sure.

Subject: Re: HTTPS?
Posted by fallingdutch on Fri, 01 Dec 2006 17:02:27 GMT
View Forum Message <> Reply to Message

zsolt wrote on Fri, 01 December 2006 13:59Is it possible in U++, to get data from secure HTTP servers?

use SSLClientSocket

Bas

Subject: Re: HTTPS?
Posted by rylek on Fri, 01 Dec 2006 18:22:31 GMT
View Forum Message <> Reply to Message

Hi there!

Honestly I have never needed a secure HTTP client myself, so I've never even checked the relevant RFC. Merely substituting normal sockets with SSL sockets should be fun using the Web/SSL module, you just need to separately download and build the SSL library couple (ssleay & libeay). In contrast to the image encoders, we haven't included these in the uppsrc plugin subtree; as they comprise about 600 source files and hosts of build and configuration options, we're afraid it would be halfway between impractical and totally insane.

Remember that to get the SSL sockets up and running, you have to somehow cope with the certificate stuff. I don't know how much support would be needed in the core client itself, it's quite likely it is possible to delegate most certificate setup & verification to the code using the HTTP client (perhaps using a callback). Please note that the Web/SSL library supports the build option .NOSSL which turns the SSL stuff off. If the HTTP client should implement support for SSL, it should also support this option to enable limiting SSL code linking to applications which really need it (the libraries have about 2 megabytes, which seems to me quite a lot).

Regards

Tomas

Subject: Re: HTTPS?
Posted by mirek on Fri, 01 Dec 2006 18:41:45 GMT
View Forum Message <> Reply to Message

BTW, this reminds me the old problem with Socket... (pick behaviour for non-container class).

One of my ideas to solve it was to simply have single Socket class both for SSL and non-SSL sockets, simply with two Open methods (Open, OpenSSL). OpenSSL would be implemented in Web/SSL.

Maybe that could help a little here?

Mirek

Subject: Re: HTTPS?
Posted by zsolt on Fri, 01 Dec 2006 19:10:21 GMT
View Forum Message <> Reply to Message

Thanks Tomas and Mirek,

I hope, in the near future I will have some time playing with HTTPS.
Quote:One of my ideas to solve it was to simply have single Socket class both for SSL and non-SSL sockets, simply with two Open methods (Open, OpenSSL). OpenSSL would be implemented in Web/SSL.

Using SSLClientSocket() function would be enough, as it can be used with normal Socket reference.
I think, the only big change in HttpClient would be the implementation of a Gate to allow accepting/refusing SSL certificates.

Subject: Re: HTTPS?
Posted by zsolt on Fri, 01 Dec 2006 19:17:03 GMT
View Forum Message <> Reply to Message

Maybe some short example would be useful, to see, what I have to do before calling SSLClientSocket().

Subject: Re: HTTPS?
Posted by mirek on Fri, 01 Dec 2006 19:23:51 GMT
View Forum Message <> Reply to Message

zsolt wrote on Fri, 01 December 2006 14:10
Using SSLClientSocket() function would be enough, as it can be used with normal Socket reference.

Yes, it is equivalent, I must have hit some intellectual low (no wonder after 20 hours fighting with gtk to get icons displayed

Still, I think the "content-transfer" aspect of Socket is quite irregular feature that should be resolved..

Mirek

Subject: Re: HTTPS?
Posted by rylek on Fri, 01 Dec 2006 19:29:38 GMT
View Forum Message <> Reply to Message

Hello,

the 'problem' with Socket implementing pick semantics is honestly more about getting Mirek to like it than a problem with the socket class as such. From the code point of view I believe the current implementation with a separate interface wrapper object and its internal socket handler object is most logical as the sockets use a completely different access interface than SSL sockets. Mixing the implementation of the two together would work well if the Open / OpenSSL methods did only some initialization of setup stuff, which they don't. The only result would be that practically every access method would have to begin with an if() distinguishing the two interfaces and wanting the

code buildable without SSL would require tens of #if-#endif pairs to mask off all these SSL variants.

Regards

Tomas

Subject: Re: HTTPS?
Posted by rylek on Fri, 01 Dec 2006 19:36:28 GMT
View Forum Message <> Reply to Message

Hi!

This is a short snippet of code I use in a commercial application to get the SSL sockets up and running. I suppose it solves a very specific situation but I hope you'll be able to use it at least to stumble upon some ideas.

Regards

Tomas

```
socket.Clear();
if(encrypt) {
 if(!ssl_context) {
  ssl_context = new SSLContext;
  if(!ssl_context->Create(SSLv3_client_method())) {
   WhenConsole(NFormat("Error creating SSL context: %s", SSLGetLastError()), 0);
   return false;
  }
  if(!IsNull(certificate_file) || !IsNull(private_key_file)) {
   String cdata = LoadFile(certificate_file);
   if(IsNull(cdata)) {
    WhenConsole(NFormat("Error reading certificate file '%s'.", certificate_file), 0);
    return false;
   }
   String pdata = LoadFile(private_key_file);
   if(IsNull(pdata)) {
    WhenConsole(NFormat("Error reading private key file '%s'.", private_key_file), 0);
    return false;
   }
   if(!ssl_context->UseCertificate(cdata, pdata)) {
    WhenConsole(NFormat("Invalid certificate '%s' / private key '%s': %s",
     certificate_file, private_key_file, SSLGetLastError()), 0);
    return false;
   }
  }
```

```
 }
 SSLClientSocket(socket, *ssl_context, host, port, true, NULL, timeout_msecs);
}
else
 ClientSocket(socket, host, port, true, NULL, timeout_msecs);
if(!socket.IsOpen()) {
 WhenConsole(NFormat(t_("Error opening socket %s:%d: %s\n"), host, port,
Socket::GetErrorText()), 0);
 return false;
}
```

Subject: Re: HTTPS?
Posted by mirek on Fri, 01 Dec 2006 19:43:44 GMT
View Forum Message <> Reply to Message

rylek wrote on Fri, 01 December 2006 14:29Hello,

the 'problem' with Socket implementing pick semantics is honestly more about getting Mirek to like it than a problem with the socket class as such. From the code point of view I believe the current implementation with a separate interface wrapper object and its internal socket handler object is most logical as the sockets use a completely different access interface than SSL sockets. Mixing the implementation of the two together would work well if the Open / OpenSSL methods did only some initialization of setup stuff, which they don't. The only result would be that practically every access method would have to begin with an if() distinguishing the two interfaces and wanting the code buildable without SSL would require tens of #if-#endif pairs to mask off all these SSL variants.

No, that is not why I am suggesting. In fact, the only change that I suggest is to transform current functions

```
bool ServerSocket(Socket& socket, int port, bool nodelay = true, int listen_count = 5, bool
is_blocking = true);
bool ClientSocket(Socket& socket, const char *host, int port, bool nodelay = true, dword *my_addr
= NULL, int timeout = DEFAULT_CONNECT_TIMEOUT, bool is_blocking = true);
bool SSLServerSocket(Socket& socket, SSLContext& ssl_context, int port, bool nodelay = true, int
listen_count = 5, bool is_blocking = true);
bool SSLClientSocket(Socket& socket, SSLContext& ssl_context, const char *host, int port, bool
nodelay = true, dword *my_addr = NULL, int timeout = DEFAULT_CONNECT_TIMEOUT, bool
is_blocking = true);
```

into methods, which would eliminate the need for exposing internal Data in Socket contructor interface. (and also would activate Assist++ hints

In fact, for backwards compatibility, I would even retain both them and that constructor...

Implementation could be as trivial as:

```
bool Socket::OpenClient(const char *host, int port, bool nodelay = true, dword *my_addr = NULL,
int timeout = DEFAULT_CONNECT_TIMEOUT, bool is_blocking = true) {
    ClientSocket(*this, host, port, nodelay, my_addr, timeout, is_blocking);
}
```

Mirek

Subject: Re: HTTPS?
Posted by rylek on Fri, 01 Dec 2006 20:07:12 GMT
View Forum Message <> Reply to Message

Hi!

I have no problem with turning the socket opening functions into methods although I don't see much that's to be gained here. The internals of the socket implementation (i.e. the Socket::Data class interface) has to be exposed at least to the Web/SSL module anyway so a reduction in the number of files used by the socket packages doesn't seem to me very likely. It would also prevent inline definition of many interface methods. The only long-term result seems to me blocking the socket class against possible future extension to other socket-like data interchange channel accessors.

Regards

Tomas

Subject: Re: HTTPS?
Posted by mirek on Fri, 01 Dec 2006 20:22:22 GMT
View Forum Message <> Reply to Message

rylek wrote on Fri, 01 December 2006 15:07Hi!

I have no problem with turning the socket opening functions into methods although I don't see much that's to be gained here. The internals of the socket implementation (i.e. the Socket::Data class interface) has to be exposed at least to the Web/SSL module anyway

Well, I consider it simply as Socket implementation divided into two packages...

Quote:
The only long-term result seems to me blocking the socket class against possible future extension to other socket-like data interchange channel accessors.

That is quite unlikely, but if that happens, we can do the same (add OpenClientFUTUREEXTENSION).

Anyway, the sole reason why I do not like the current state is: If socket has PIMPL exposed in interface, why Stream does not have the same design? If we add this to the Stream, why not Ctrl?

Mirek

Subject: Re: HTTPS?
Posted by Weras on Mon, 31 Aug 2009 21:13:51 GMT
View Forum Message <> Reply to Message

Hi all!

Sorry my english, but i have no idea, where else i can find the solution to my problem.

I want to use SSL connection to server in my project. I downloaded opensll and try to setup them on my pc. But i could not get libeay32.dll & libeay32.dll.

Can you tell how to properly and fully installed openssl to u++? May be there is some link to resource?

Thx.

Subject: Re: HTTPS?
Posted by rylek on Wed, 02 Sep 2009 21:18:33 GMT
View Forum Message <> Reply to Message

Hello!

So far I've always downloaded the OpenSSL source package from www.openssl.org and used it to build the libraries. Remember that normally it's easier to link the SSL libraries statically to the final application because then you don't have to worry about search paths or different versions of the (very frequently used) SSL libraries; in such case you would need no dll's at all, only the lib's. However, all three versions (DLL version, single-threaded and multi-threaded statically linked version) can be built using some options of the OpenSSL build script. Remember that in order to build OpenSSL you need three main things:

1) working installation of a compiler, I built it under MSC but MinGW should be fine

2) an assembler installed (MASM or NASM)

3) the Perl interpreter; this can be downloaded and installed separately, certain software packages install it automatically (like the Oracle server).

After you manage to make the build scripts produce the "libeay32.lib" and "ssleay32.lib" in the out32 / out32dll / out32mt output directories, to complete OpenSSL "installation" under U++ it should be sufficient to add the include path (<openssl installation directory>/include) and the library path (out32 or variants) to the respective path lists in your desired build method definition.

If you fail to complete the above described process, please write how far you managed to get and I'll try to describe the relevant step in greater detail.

Regards

Tomas
