Subject: SFTP or full SSH2 support for U++?
Posted by Oblivion on Sun, 10 Jan 2016 11:58:48 GMT
View Forum Message <> Reply to Message

Hello,

Some time ago when I uploaded my FTPS implementation to the bazaar section, Daniel (Unodgs) asked me If I have any plans to implement an SFTP class.
And I'd said yes.

Now I am here to deliver my promise. :)

But before I upload the package to the bazaar section, I have to make a choice and would like to hear your opinion on the subject.

First of all, I decided not to implement the SSH2 protocol from the scratch. I could've, but it would be a) tedious and b) wasting my time.
So I decided searching for alternatives.
The best alternative that I could have come up with was wrapping an existing, multiplatform (Linux and Windows, at least) library.
I found out that libssh2 was the way to go. It was plain C, easy to wrap up and had both blocking and non-blocking modes of operation available.

I finally implemented an SFTP class for U++ using libssh2, which has the below features:

1) Allows blocking and non-blocking operations. Namely, it can work in both synchronous and asynchronous modes.
I use HttpRequest and NetworkProxy class' asynchronous design here too, since that design proved very effective.

NetworkProxy class.

3) Takes advantage of U++ streams in file upload and download operations, and uses gates for progress tracking when needed.

4) Supports both MD5 and SHA 1 methods.

5) Has SFtp::DirEntry class for easy parsing of directory entries.

In short, currently we have a sync/async SFTP wrapper that can authenticate using "public key" method and/or Username/password combination, and perform Open/Close/Read/Write/List/Rename/Delete operations on remote file system entries. Also its API is very similar to my FTPS class, and its design derives from NetworkProxy and HttpRequest classes and follows the U++ coding style.

Now, the class is abstracted nicely so that I can either let it be a standaloune, robust SFtp class, or take it one step further and wrap up the whole functionality of libssh2.
Latter means adding SSHSession, SSHChannel, Scp classes to the package too.
What is your opinion?

(For example I can also easily add the SCP protocol to the package. SCP is known to perform better in some type file read/wtire operations.)

Also, since the libssh2 uses its own allocators (alloc, realloc, dealloc) but also gives the user the choice to implement his/her own, I'll definietly need help here.
I'm not experienced with U++ allocators.

Here is an actual, barebone SFtp example, demonstrating directory listing and file download, using a public SFtp test server:

```
#include <Core/Core.h>
#include <SFTP/SFTP.h>

using namespace Upp;

static const char* CRLF = "\r\n";

// This callback allows getting directory entries one by one.
// It is optional.
bool ListDirectory(SFtp::DirEntry& e)
{


 String ls = e.GetEntry();
 if(!ls.IsEmpty())
  // When available, traditiona UNIX style directory listing can be used.
  Cout() << e.GetEntry() << CRLF;
 else {
  Cout()  << e.GetName()          << CRLF
    << e.GetUserID()  << CRLF
    << e.GetGroupID()           << CRLF
    << e.GetSize()    << CRLF
```

```
    << e.GetLastAccessed()        << CRLF
    << e.GetLastModified()        << CRLF
    << "Permissions:"  << CRLF
    << "R: " << e.IsReadable()  << ", "
    << "W: " << e.IsWritable()  << ", "
    << "X: " << e.IsExecutable()    << CRLF;
 }
 return false;
}

CONSOLE_APP_MAIN
{
 // This example demonstrates directory listing (ls) and file download operations,
 // using a public SFTP test server.
 // Note that this example uses blocking mode. It is synchronous.
 // There is also a non-blocking, asynchhronous mode.

 FileOut wallpaper("/home/genericuser/Wallpapers/SFTP-download-test-wallpaper.jpg");
 SFtp::DirList dirlist; // SFtp::DirList is a vector containing SFtp:DirEnty elements.

 // Enable logging.
 SFtp::Trace();

 SFtp sftp;
 if(sftp.User("demo-user", "demo-user").Connect("demo.wftpserver.com", 2222)) {
  // Get server banner.
  Cout() << sftp.GetBanner() << CRLF;
  if(sftp.OpenDir("/download") && sftp.ListDir(dirlist, callback(ListDirectory))) {
   if(sftp.Open("/download/F11_wallpaper_06_1600x1200.jpg", SFtp::READ)) {
    if(sftp.Get(wallpaper)) {
            Cout() << "File successfully downloaded.\r\n";
                        return;
                    }

   }
            }
 }
 Cout() << "Failed.\r\n" << "Reason: " << sftp.GetErrorDesc() << CRLF;

 // Instead we could have used the SFtpGet() convenience function.
 // int rc = SFtpGet(wallpaper, "demo-user", "demo-user", "demo-wftpserver.com", 2222,
"/download/F11_wallpaper_06_1600x1200.jpg");

}
```

If Sftp is sufficient, I'll upload it to the bazaar next weekend. If not, I'l still add it to the bazaar next

weekend but implement the missing classes incrementally over time (next class to implemnt will be SCP). :)

Regards,

Oblivion

Subject: Re: SFTP or full SSH2 support for U++?
Posted by mirek on Sun, 10 Jan 2016 22:02:37 GMT
View Forum Message <> Reply to Message

Hi,

sounds good.

One thing this check is the license. We generally prefer BSD licensed stuff.

As for what to add, SFTP is great, but adding SSH support would be just awesome.

Do no be afraid about allocators, see e.g. Core/SSL/InitExit.icpp, I bet things will be pretty similar with SFTP.

Mirek

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Mon, 11 Jan 2016 08:10:29 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Thanks for the tip, I'll look into Core/SSL/InitExit.icpp

Ok then, next week I'll upload the Sftp, and then go for full SSH2 support in general, incrementally adding methods and classes to the package.
(There are and will be some design decisions and trade-offs, though.)

As to the license, I'd considered that before chosing libssh2. It has a revised BSD license (http://www.libssh2.org/license.html)
Hope this is ok. :)

Regards,
Oblivion

Subject: Re: SFTP or full SSH2 support for U++?
Posted by mirek on Mon, 11 Jan 2016 08:47:23 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Mon, 11 January 2016 09:10
As to the license, I'd considered that before chosing libssh2. It has a revised BSD license (http://www.libssh2.org/license.html)
Hope this is ok. :)

Perfect! :)

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Tue, 26 Jan 2016 20:02:12 GMT
View Forum Message <> Reply to Message

Hello,

Just to give you all an update:

U++ wrapper for libssh2 is in good shape.

SSHSession class: implemened. I borrowed the U++ memory allocation code (re/alloc, free) from Core/SSL package and after a slight modification glued it to the SSH package. (It is working!)

Sftp class: Implemented all the necessary commands (seek/seek64/statvfs not yet implemented).

I don't have access to Windows right now, so the code is currently tested only on an up-to-date arch linux installation (with a KDE/Plasma5 desktop).

Below is a screenshot of a very simple (and easy to write) sftp downloader example with 10 concurrent downloads, demonstrating non-blocking/async operation capabilities.

Below is the actual code responsible for 10 concurrent downloads, demonstrating sftp basic async api (Some parameters are hard coded. I was being lazy.)

```
// Async jobs.

struct Job {
 SFtp  sftp;
 FileOut file;
```

```
 String  path;
 int  index;
 int  cmd;
};

enum Command { OPEN, READ, CLOSE, FINISH };

const char *sftp_server = "demo.wftpserver.com";
const char *sftp_user   = "demo-user";
const char *sftp_pass   = "demo-user";
const char *remote_file = "/download/F11_wallpaper_06_1600x1200.jpg";

void SFtpExample::Download()
{
   // Initialize and fill an array of Job(s)

   for(int i = 0; i < 10; i++) {
      Job& job = jobs.Add();
      job.sftp.User(sftp_user, sftp_pass);
      job.sftp.StartConnect(sftp_server, 2222);
      job.sftp.WhenDo = THISBACK(UpdateGui);
      job.file.Open(Format("/home/testuser/picture-%d.jpg", i));
      job.cmd     = OPEN;
      job.index   = i;
   }

   // Actual loop: connects to the server and downloads a file in a concurrent way.

   while(jobs.GetCount()) {
      for(int i = 0; i < jobs.GetCount(); i++) {
         Job& job = jobs[i];
         SocketWaitEvent e;
         e.Add(job.sftp.GetSocket());
         e.Wait(10);
         job.sftp.Do();
         if(!job.sftp.InProgress()) {
            if(job.sftp.IsSuccess()) {
               switch(job.cmd) {
                  case OPEN:
                     job.sftp.StartOpen(remote_file, SSH::READ);
                     job.path = remote_file;
                     job.cmd  = READ;
                     continue;
                  case READ:
                     job.sftp.StartGet(job.file, THISBACK2(DownloadProgress, job.index, job.path));
                     job.cmd  = CLOSE;
                     continue;
                  case CLOSE:
```

```
                    job.sftp.StartClose();
                    job.cmd  = FINISH;
                    continue;
                case FINISH:
                    break;
            }
        }
        else
        if(job.sftp.IsFailure())
            list.Set(job.index, 1, DeQtf(job.sftp.GetErrorDesc()));
        jobs.Remove(i);
        list.Remove(i);
        for(int n = 0; n < jobs.GetCount(); n++)
            jobs[n].index = n;
    }
  }
 }
}
```

P.s. I delayed the upload of the package, since it still has some rough edges to iron-out.

Regards,
Oblivion

File Attachments

1) SFTP Concurrent Downloads.jpeg, downloaded 1580 times

Subject: Re: SFTP or full SSH2 support for U++?
Posted by mirek on Sun, 31 Jan 2016 08:26:25 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Tue, 26 January 2016 21:02Hello,

Just to give you all an update:

U++ wrapper for libssh2 is in good shape.

SSHSession class: implemened. I borrowed the U++ memory allocation code (re/alloc, free) from Core/SSL package and after a slight modification glued it to the SSH package. (It is working!)

Sftp class: Implemented all the necessary commands (seek/seek64/statvfs not yet implemented).

I don't have access to Windows right now, so the code is currently tested only on an up-to-date arch linux installation (with a KDE/Plasma5 desktop).

Below is a screenshot of a very simple (and easy to write) sftp downloader example with 10 concurrent downloads, demonstrating non-blocking/async operation capabilities.

Below is the actual code responsible for 10 concurrent downloads, demonstrating sftp basic async api (Some parameters are hard coded. I was being lazy.)

```
// Async jobs.

struct Job {
 SFtp  sftp;
 FileOut file;
 String  path;
 int  index;
 int  cmd;
};

enum Command { OPEN, READ, CLOSE, FINISH };

const char *sftp_server = "demo.wftpserver.com";
const char *sftp_user   = "demo-user";
const char *sftp_pass   = "demo-user";
const char *remote_file = "/download/F11_wallpaper_06_1600x1200.jpg";

void SFtpExample::Download()
{
   // Initialize and fill an array of Job(s)

   for(int i = 0; i < 10; i++) {
      Job& job = jobs.Add();
      job.sftp.User(sftp_user, sftp_pass);
      job.sftp.StartConnect(sftp_server, 2222);
      job.sftp.WhenDo = THISBACK(UpdateGui);
      job.file.Open(Format("/home/testuser/picture-%d.jpg", i));
      job.cmd     = OPEN;
      job.index   = i;
   }

   // Actual loop: connects to the server and downloads a file in a concurrent way.

   while(jobs.GetCount()) {
      for(int i = 0; i < jobs.GetCount(); i++) {
         Job& job = jobs[i];
         SocketWaitEvent e;
         e.Add(job.sftp.GetSocket());
```

```
            e.Wait(10);
            job.sftp.Do();
            if(!job.sftp.InProgress()) {
               if(job.sftp.IsSuccess()) {
                  switch(job.cmd) {
                     case OPEN:
                        job.sftp.StartOpen(remote_file, SSH::READ);
                        job.path = remote_file;
                        job.cmd  = READ;
                        continue;
                     case READ:
                        job.sftp.StartGet(job.file, THISBACK2(DownloadProgress, job.index, job.path));
                        job.cmd  = CLOSE;
                        continue;
                     case CLOSE:
                        job.sftp.StartClose();
                        job.cmd  = FINISH;
                        continue;
                     case FINISH:
                        break;
                  }
               }
               else
               if(job.sftp.IsFailure())
                  list.Set(job.index, 1, DeQtf(job.sftp.GetErrorDesc()));
               jobs.Remove(i);
               list.Remove(i);
               for(int n = 0; n < jobs.GetCount(); n++)
                  jobs[n].index = n;
            }
         }
      }
   }
```

P.s. I delayed the upload of the package, since it still has some rough edges to iron-out.

Regards,
Oblivion

Cool. Yesterday I had to fix some ugly PHP application (do not ask...) on one of those hosting platforms where you only have SFTP access.

I was thinking that it would be a cool demonstration (and a very nice PR CodeProject article) to create actual EDITOR of text files over FTP/FTPS/SFTP in "list of files in the left" style. Not sure I will have enough time for that, but it would be fun.

Mirek

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Sun, 31 Jan 2016 23:06:24 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Quote:
I was thinking that it would be a cool demonstration (and a very nice PR CodeProject article) to create actual EDITOR of text files over FTP/FTPS/SFTP in "list of files in the left" style. Not sure I will have enough time for that, but it would be fun.

Writing an FTP(S) & SFTP editor wouldn't be too difficult. We can use Any container and dynamically create both. Plus, directory enty parsers of the both classes are very similar. It would be easy to achive some abstract API when writing the editor. What worries me is that it'll end up in a somewhat complex code, and that might be bad for PR. Instead, we can write a UI skeleton and build two separate editors, one utilizing FTP(s), and the other SFTP. In this way, people can study the code(s) and the differences. While I am currently polishing SSH/SFtp package, I can start writing a basic FTPS based text editor, in the meantime.

By the way, I was going to propose writing a CodeProject article for TcpSocket (I can write it in the following weeks, bu you may need to review it first :)). IMO, it is a vital part of Upp/Core. Yet it is somewhat an enigma. It is well designed but poorly documented. (not talking about its API doc). I had to study HttpRequest/GuiWebCrawler to understand how to do nonblocking operations. Maybe we can clarify that.
Also I would like to propose adding two methods for TcpSocket: My experiences with sockets and remote operations showed me that sending local IP over sockets is not uncommon.
So, maybe it is possible to add a counterpart to GetPeerAddr() method: GetLocalAddr()?
And of course EWOULDBLOCK is technically an error, but in practice we don't always treat it that way.
So I believe adding a public WouldBlock() method to check if the socket would block would be helpful in writing nonblocking code. (GetError() is not very helpful when reading and studying the source code).

Regards,
Oblivion.

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Sun, 06 Mar 2016 22:39:46 GMT
View Forum Message <> Reply to Message

Hello guys,

A major update:

SSH package is re-written using the new version of AsyncQueue class which can utilize ordinary callbacks and lambdas, instead of the cumbersome VarArgs class. SSH package relies heavily on lambda functions, therefore it will require C++11. I believe it is much more elegant now, since it drastically simplifed the task and reduced the source code significantly. It is almost complete, and first release is at hand. (SFtp component, at least).

Below code snippet is taken from the new version of a simple download test app that concurrently downloads a number of jpg files from a well-known public Sftp test server. As you can see, unnecessary state tracking is removed:

```
struct Job {
	SFtp    sftp;
	FileOut file;
	int     index;
};

Array<Job> jobs;


//.....

const char *sftp_server     = "demo.wftpserver.com";
const char *sftp_user       = "demo-user";
const char *sftp_password   = "demo-user";
const char *remote_path     = "/download";

bool SFtpDownloader::ReadDir(Ssh::DirList& ls)
{
   // We'll get the directory listing, using a synchronous (blocking) call.

   SFtp browser;
   browser.WhenDo << [&] {ProcessEvents(); };                                  // GUI should be
responsive...
   if(browser.Connect(sftp_server, 2222, sftp_user, sftp_password)) {

      // Let us use a convenience method which works on paths, not file descriptors.

      if(browser.ListDir(remote_path, ls))
         return true;
   }
   Exclamation(browser.GetErrorDesc());
   return false;
}
```

```
void SFtpDownloader::Download()
{
   // Wait for the queue to be processed.

  if(!jobs.IsEmpty())
      return;

   Ssh::DirList ls;
   if(!ReadDir(ls))
      return;
   for(int i = 0, j = 0; i < ls.GetCount(); i++) {

      // We can work on directory entries easily.

      Ssh::DirEntry& e       = ls[i];
      const Ssh::Attrs& attrs = e.GetAttrs();

      // Let us simply try to download all files with *.jpg extension to the current directory,
asynchronously.
      // Here, thanks to AsyncQueue, asynchronous calls work in a fire-and-forget fashion. Similar
to "batch processing".
      // E.g., any number of Sftp commands/jobs can be queued to be processed without
intervention (till they're finished, or failed, of course).

      if(attrs.IsFile() && GetFileExt(e.GetName()).IsEqual(".jpg")) {
         Job& job    = jobs.Add();
         job.index   = j;
         job.file.Open(Format("%s/SFtpExample-%d-%s", GetHomeDirectory(), job.index,
e.GetName()));
         job.sftp.StartConnect(sftp_server, 2222, sftp_user, sftp_password);
         job.sftp.StartGet(job.file, Format("%s/%s", remote_path, e.GetName()), Ssh::READ, 0755,
THISBACK2(DownloadProgress, job.index, e.GetName()));
         j++;
      }
   }

   //  Below loop does asynchronous job processing.

   while(!jobs.IsEmpty()) {
      int i = 0;
      SocketWaitEvent we;
      we.Add(jobs[i].sftp.GetSocket());
      we.Wait(10);
      while(i < jobs.GetCount()) {
         SFtp& sftp = jobs[i].sftp;
         sftp.Do();
         if(!sftp.InProgress()) {
```

```
            // For the sake of simplicity, we do not differentiate between success and failure.
            jobs.Remove(i);
            break;
        }
        ProcessEvents();
        i++;
    }
  }
}

//.....
```

Before I release the first public [beta] version, I'll have to write its docs. It'll probably happen this week.

Regards,

Oblivion.

File Attachments

1) Sftp - New version.jpeg, downloaded 1463 times

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Tom1 on Sat, 13 May 2017 19:01:50 GMT
View Forum Message <> Reply to Message

Hi Oblivion,

This sounds interesting and might well fit my needs to automate some file transfers. Is it possible to get a hold of a copy of this SFTP client module of yours?

Best regards,

Tom

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Sun, 14 May 2017 14:43:54 GMT
View Forum Message <> Reply to Message

Quote:Hi Oblivion,

This sounds interesting and might well fit my needs to automate some file transfers. Is it possible to get a hold of a copy of this SFTP client module of yours?

Best regards,

Tom

Hello Tom,

Since you've revived this topic, let me give some information on recent developments. :)

Some time ago I decided to go for a full SSH package. As of today, SFtp and Scp modules are complete, and Exec module is to-be-written.
I'm cleaning up the codebase now. A technical preview version is scheduled, and will be available in June 10, 2017. :)
I'll publish and further develop it for U++ with a BSD license, so of course you'll have the chance to get a copy.
But I'm afraid you'll have to wait a little more.

Regards,

Oblivion

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Tom1 on Sun, 14 May 2017 15:46:48 GMT
View Forum Message <> Reply to Message

Hi Oblivion,

Excellent! I'll stay tuned for the technical preview on this channel... :)

If I'm not reacting to your technical preview announcement in a reasonable time, please PM me about the opportunity to test it.

Best regards,

Tom

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Tue, 27 Jun 2017 23:13:27 GMT
View Forum Message <> Reply to Message

Hello everyone,

It is now time to publish the technical preview version of SSH package for Upp. :)

A brief status of the project:

SSH package for U++

SSH package is a libssh2 wrapper for Ultimate++.
It aims to be a complete secure shell client solution. (SFtp, Scp, Exec, Terminal, X11Exec, KnownHosts)
Currently it is at a technical preview stage, which means it is a work-in-progress, and subject to change.

Requirements:

- A C++ compiler with at least C++11 support.
- libssh2 ssh library (https://www.libssh2.org/)

Features and Highlights:

- Tight integration with U++.
- Support for both synchronous and asynchronous operations with both high level and low level methods.
- Allows using U++ style memory [de/re]allocators with libssh2.
- Ssh subsystems support RTTI, and share a common interface for polymorphism.
  Namely, it is possible to store different subsystems in the same U++ container.
  (e.g. SFtp, Exec, Scp, Shell can be all stored in the same Array.)

Todo:

- Add known hosts and ssh agents support.
- Add more high level methods.
- Add ssh X11 support.
- Ship libssh2 library with the package.
- Documentation.

History:

2017-06-28: EAGAIN is now properly handled across the subsystems.
        WhenWrite and WhenRead callbacks are removed in favour of parametrized gates.
2017-06-27: SFtp::StartStop: fixed.

2017-06-27: CleanUp() methods are added to the classes Ssh, SFtp, and Channel.
	This method utilizes JobQueue's WhenCleanup event.
2017-06-26: U++ memory [de/re]allocators added and made default.
	initial support for ssh shell, and terminal added.
	Subsystems now perform a clean up on failure to prevent possible heap leaks.
2017-06-22: Initial release.

At the moment I am documenting the Package and its api.
There are some important points you need to know before getting yourself familiar with the code.

As I've mentioned in the above summary, SSH package is a libssh2 wrapper. But currently it does not contain libssh2 source code (which also has BSD license).
On linux, your distribution probably provides the lib. On Windows you'll need cmake for express compilation.

Design:

The classes of SSH package are based on JobQueue class, a simple job-queue model (a callback queue based on Upp::Vector actually).
JobQueue is meant to work with (basically) sockets. It provides a uniform interface for async socket operations. In fact, it is an abstraction of Upp's own HttpRequest class' async interface.
From my experience, I can say that it is pretty scalable with a negligable overhead.It allows batch processing (e.g. you can queue the commands and then batch-process them later.)
However if you are going to batch process some commands, do note that the queue will treat the batch as a whole. Hence a failure in one command will halt and clear the queue. This is a security measure to prevent any misbehaviour.

Other advantage of JobQueue is that it allows writing code in a uniform coding pattern. (You can see it in SSH package clearly), in the sense that it exposes both a small set of common functions, and a basic structure.
All SSH sub/classes are based on JobQueue. Therefore they expose a uniform asynchronous api.

Also, each blocking method has its nonblocking counterpart (starting with "Start" prefix.)

Categorization:

SSH package is divided into several classes, based on system-subsystems relationship.
[
Ssh		-> Main session class (system). Provides connection and authentication mechanism. Although known Hosts support not added yet, it will be available in the following versions.

SshSubsystem -> SshSubsystem is base of all SSH protocol based communication subsystems.

SFtp		-> Complete, but I'm planning to further enhance it with more high level commannds, once the api stabilized.

Channel -> Mostly done.
Scp -> Complete.
Exec -> Complete.
Shell -> Not added yet. Besic support is present. And will be developed durther.
Terminal -> Not added yet. Basic support is present. And will be developed further.
X11Exec -> Not added yet. Will be added incrementally.

Core class, as one might expect is, Ssh. :)
It provides connection and authentication mechanisms.

All other Ssh subsystems are derived from a single base class: SshSubsystem.
This class provides common operations and connects subsystems to the Ssh session.

Also it has RTTI interface, which brings us to polymorphism:
SSH package allows a great level of polymorphism while keeping the crucial interface uniform throughout the derivative classes (thanks to JobQueue).
Hence it is possible to put, say, a Scp, Exec, Sftp, Terminal in a single Upp::Array and execute them asynchoronusly through a uniform calling pattern.

All other classes -except SFtp- are derived directly from Channel subsystem. All derivatives expose Channel commands, while has their own commands. So any Channel-based derivative use the power of Channel.

Memory Allocation:

libssh2 provides support for custom allocators. (or alternatively it can use system's memory allocators via USEMALLOC flag)
Basically I copied and adjusted the memory manager found in Core/SSL package.
It works fine with libssh2, but I'm not very happy with it (code duplication is something I don't like)
Maybe there is a better way?

How To:

Package currently contains three simple examples:

SshExec: This simple example demonstrates a remote command execution via ssh2 protocol's exec channel, and the blocking api of SSH package.
SFtpDir: This simple example demonstrates a remote diretory listing via sftp channel, and the blocking api of SSH package.
SshAsyncRequests: This simple example demonstrates polymorphism with ssh subsystems, and asynchronous calls, combining a SFtp and Exec instance.

In fact, the first two example are so simple that they look like little code snippets: :)

SshExec:

```
#include <Core/Core.h>
#include <SSH/SSH.h>

using namespace Upp;

// SshExec.cpp:
// Demonstrates remote command execution on an ssh channel.

// A popular public test server:
// ----------------------------
// Host:     test.rebex.net
// User:     user
// Password: password
// Command:  ls -l

CONSOLE_APP_MAIN
{
 Ssh session;
 Cout() << "Hostname: ";
 auto host = ReadStdIn();
 Cout() << "Username: ";
 auto user = ReadStdIn();
 Cout() << "Password: ";
 auto pass = ReadStdIn();
 const int port = 22;

 if(session.Connect(host, port, user, pass)) {
  Cout() << "Command: ";
  auto cmd = ReadStdIn();
  Exec xc(session);
  auto rc = xc(cmd, Cout(), Cerr());
  if(xc.IsFailure()) Cerr() << Format("Exec failed. %s", xc.GetErrorDesc());
  else Cout() << Format("Remote program exited with code: %d\n", rc);
 }
 else Cerr() << session.GetErrorDesc() << '\n';
}
```

Or, SFtpDir:

```
#include <Core/Core.h>
#include <SSH/SSH.h>

using namespace Upp;

// SFtpDir.cpp:
// Demonstrates remote directory listing via an sftp channel.
```

```
// A popular public test server:
// ----------------------------
// Host:     test.rebex.net
// User:     user
// Password: password
// Path:     pub/examples

CONSOLE_APP_MAIN
{
 Ssh session;
 Cout() << "Hostname: ";
 auto host = ReadStdIn();
 Cout() << "Username: ";
 auto user = ReadStdIn();
 Cout() << "Password: ";
 auto pass = ReadStdIn();
 Cout() << "Path: ";
 auto path = ReadStdIn();
 const int port = 22;

 if(session.Connect(host, port, user, pass)) {
  SFtp sftp(session);
  SFtp::DirList ls;
  if(!sftp.ListDir(path, ls)) Exit(sftp.GetError());
  for(auto& e : ls)  Cout() << e.GetEntry() << "\n";
 }
 else Cerr() << session.GetErrorDesc() << '\n';
}
```

That's it :)

Please keep in mind that while ths package is reasonably stable it is still a work-in-progress. So, USE IT AT YOUR OWN RISK. :)

Although, given my personal experience I am pretty satisfied by SSH package's api, it is by no means set in stone. It'll change when necessary.
I'll be grateful for any criticism, suggestions, reviews, patches, etc. After all, this package is for U++ and its users.

Package tested on Win7/10 (Mingw64), and latest Arch Linux and Fedora (25).

p.s. There is a leftover that must be changed, which I realized this morning. (In SshSubsystem::To() method) I will change it tonight, but in the meanwhile if you are going to test the package please consider changing the reinterpret_cast to dynamic_cast.

Cheers,
Oblivion.

File Attachments

1) SSH (Technical Preview and Examples).zip, downloaded 399 times

---

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Tom1 on Wed, 28 Jun 2017 12:46:22 GMT
View Forum Message <> Reply to Message

Hi Oblivion,

This looks good. However, I may need to wait for the following item on your task list to be completed:

- Ship libssh2 library with the package.

Compiling libssh2 became a bit frustrating on my Windows system, so I decided to rather wait for libssh2 becoming part of the delivery. In my view, the libssh2 should be embedded inside your SSH package in a way that the whole package just gets built and statically linked in Windows like any other package without dependencies. Well, I guess the dependency of Core/SSL should be there to make it work.

Best regards,

Tom

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Wed, 28 Jun 2017 13:04:38 GMT
View Forum Message <> Reply to Message

Quote:Re: SFTP or full SSH2 support for U++? Wed, 28 June 2017 15:46
Tom1
Hi Oblivion,

This looks good. However, I may need to wait for the following item on your task list to be completed:

- Ship libssh2 library with the package.

Compiling libssh2 became a bit frustrating on my Windows system, so I decided to rather wait for libssh2 becoming part of the delivery. In my view, the libssh2 should be embedded inside your SSH package in a way that the whole package just gets built and statically linked in Windows like any other package without dependencies. Well, I guess the dependency of Core/SSL should be there to make it work.

Best regards,

Tom

Hello Tom,

Of course I am going to include the libssh2 package in the following days (probably next week). :)
There's too much frustration about libssh2's windows build. And those complaints are usually right.
The problem is that the libssh2 team didn't seem to provide a clear compilation guide.
While I personally prefer OpenSSL, it is not mandatory for libssh2. Windows has CNG since Vista. I can make it an option too.

Fortunately It is actually very easy to compile libssh2 on windows. (which makes you get even more frustrated, finding this after hours of of code-digging).

In the mean time I can write a step by step guide (with screenshots) on how to compile it on Windows (using mingw64, with optional OpenSSL or WinCNG).

Best regards,
Oblivion.

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Tom1 on Wed, 28 Jun 2017 13:29:43 GMT
View Forum Message <> Reply to Message

Hi Oblivion,

Thanks, but you do not need to write the libssh2 compilation guide for my sake. I'm not in that much hurry with this and I can certainly wait a little bit longer so that you can merge the libssh2 inside a U++ package.

BTW: I wish this package is then compilable with MSC14 and above, since I cannot use mingw because of some special external library dependencies I have.

Best regards,

Tom

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Wed, 28 Jun 2017 13:50:27 GMT
View Forum Message <> Reply to Message

AFAIK Koldo (who had the chance to test my code first hand, and whom I should thank here!) has managed to compile it on VS 2015.
IT needs slight changes (such as that some "."s in lambda functions should be substituted with "->"s), which I'll make this week end.
The thing is, I do not use MSC. I guess I'll install it as a testing environment. :)

Best regards,
Oblivion

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Tom1 on Wed, 28 Jun 2017 14:04:00 GMT
View Forum Message <> Reply to Message

OK. Keep up the good work! :)

Best regards,

Tom

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Fri, 30 Jun 2017 21:38:30 GMT
View Forum Message <> Reply to Message

Hello,

Technical preview version of the SSH package is updated. Now it contains the libssh2 library. And uses the OpenSSL supplied by U++. :)

I've tested it with GCC on Linux, and Mingw64 on Windows. It is compiled successfully on both systems.
I have yet to test it with MSC.

Please test it.

If it fails to compile, try configuring the libssh2 library, using libssh2_config.h header.
That header file contains defs. Should you need to modify it, possible values can be copied from the libssh2/libssh2_config.h.in file.

Maybe I should open a seperate topic on SSH package in the Bazaar section of the forums for the time being?

Suggestions, patches, and cricisim is welcome.

[See below post for the updated package]

Best regards,
Oblivion

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Tom1 on Sat, 01 Jul 2017 15:35:39 GMT
View Forum Message <> Reply to Message

Hi Oblivion,

Thanks! Here are the quick results:

I tried to compile your packages on Windows 10 Pro 32 bit with MSC14. I just copied all your packages to bazaar (locally). (Running on top of upp rev. 10804.)

When compiling, I got the following errors and warnings for SSH:

----- SSH (MSC14 MSC WIN32) (2 / 6)
Ssh.cpp
C:\upp-10804\bazaar\SSH\Ssh.cpp(28): error C2143: syntax error: missing ';' before '.'
C:\upp-10804\bazaar\SSH\Ssh.cpp(28): error C2059: syntax error: '.'
C:\upp-10804\bazaar\SSH\Ssh.cpp(35): error C2059: syntax error: '.'
C:\upp-10804\bazaar\SSH\Ssh.cpp(46): error C2143: syntax error: missing ';' before '.'
C:\upp-10804\bazaar\SSH\Ssh.cpp(46): error C2059: syntax error: '.'
Channel.cpp
C:\upp-10804\bazaar\SSH\Channel.cpp(134): warning C4244: '=': conversion from 'Upp::int64' to 'int', possible loss of data
SFtp.cpp
C:\upp-10804\bazaar\SSH\SFtp.cpp(269): warning C4244: 'argument': conversion from 'Upp::int64' to 'int', possible loss of data
C:\upp-10804\bazaar\SSH\SFtp.cpp(269): warning C4244: 'argument': conversion from 'Upp::int64' to 'std::size_t', possible loss of data
C:\upp-10804\bazaar\SSH\SFtp.cpp(277): warning C4554: '&': check operator precedence for possible error; use parentheses to clarify precedence
c:\upp-10804\bazaar\ssh\ssh.h(165) : error C4716: 'Upp::SFtp::StartGetStat': must return a value
c:\upp-10804\bazaar\ssh\ssh.h(174) : error C4716: 'Upp::SFtp::StartRealPath': must return a value
c:\upp-10804\bazaar\ssh\sftp.cpp(231) : error C4716: 'Upp::SFtp::StartListDir': must return a value
Exec.cpp
Scp.cpp
libssh2upp.c
c:\upp-10804\bazaar\ssh\libssh2\libssh2_config.h(105): error C2375: 'snprintf': redefinition; different linkage

c:/program files/windows kits/10/include/10.0.14393.0/ucrt\stdio.h(1932): note: see declaration of 'snprintf'
c:\upp-10804\bazaar\ssh\libssh2/agent.c(300): warning C4996: 'sprintf': This function or variable may be unsafe. Consider using sprintf_s instead. To disable de
    precation, use _CRT_SECURE_NO_WARNINGS. See online help for details.
c:/program files/windows kits/10/include/10.0.14393.0/ucrt\stdio.h(1769): note: see declaration of 'sprintf'
c:\upp-10804\bazaar\ssh\libssh2/channel.c(43): fatal error C1083: Cannot open include file: 'unistd.h': No such file or directory
Malloc.cpp
SSH: 7 file(s) built in (0:09.19), 1313 msecs / file, duration = 9515 msecs, parallelization 100%

There were errors. (1:40.53)

I have no time to look at the causes immediately, but hope these help you a bit. :)

When I get to office, I will try on a 64-bit Windows platform.

Best regards,

Tom

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Sun, 02 Jul 2017 15:20:41 GMT
View Forum Message <> Reply to Message

Thank you very much for providing feedback, Tom!

I've installed and configured the MSC to work with TheIDE.
It now compiles on VS 2015. Hopefully the errors should be fixed now.

Tested on:

Windows: MINGW 32/64, MSC (2015)
Linux: GCC

Please test it.

Best regards,

Oblivion.

File Attachments
1) SSH Package (technical preview).zip, downloaded 363 times

Subject: Re: SFTP or full SSH2 support for U++?
Posted by koldo on Sun, 02 Jul 2017 17:14:18 GMT
View Forum Message <> Reply to Message

Dear Oblivion

Thank you very much. It works again :), and with libssh2 embedded!

Just a couple of details (sorry for my perfectionism :p):

- Old scp demo does not work

```
Cout() << "Getting file Demo.mo\n";
FileOut fout(AppendFileName(GetDesktopFolder(), "compile_run.sh"));
const char *path = "compile_run.sh";
Scp scp(ssh);
scp.WhenRead  = [=](int64 total, int64 done) { return false; };
scp.WhenWrite = Proxy(scp.WhenRead);
if(!scp.Get(fout, path))
 Cout() << scp.GetErrorDesc();
fout.Close();
```

- SFtp::Init(), Stop(), and Ssh.WhenWait() do not exist.

- Some warnings got by MSC15-64

```
SFtp.cpp
C:\upp\bazaar\SSH\SFtp.cpp(102): warning C4267: 'argument': conversion from 'size_t' to
'unsigned int', possible loss of data
C:\upp\bazaar\SSH\SFtp.cpp(136): warning C4267: 'argument': conversion from 'size_t' to
'unsigned int', possible loss of data
C:\upp\bazaar\SSH\SFtp.cpp(151): warning C4267: 'argument': conversion from 'size_t' to
'unsigned int', possible loss of data
C:\upp\bazaar\SSH\SFtp.cpp(181): warning C4267: 'argument': conversion from 'size_t' to
'unsigned int', possible loss of data
C:\upp\bazaar\SSH\SFtp.cpp(237): warning C4267: 'argument': conversion from 'size_t' to
'unsigned int', possible loss of data
C:\upp\bazaar\SSH\SFtp.cpp(252): warning C4267: 'argument': conversion from 'size_t' to
'unsigned int', possible loss of data
C:\upp\bazaar\SSH\SFtp.cpp(269): waring C4267: 'argument': conversion from 'size_t' to 'int',
possible loss of data
C:\upp\bazaar\SSH\SFtp.cpp(269): warning C4244: 'initializing': conversion from 'ssize_t' to 'int',
possible loss of data
C:\upp\bazaar\SSH\SFtp.cpp(300): warning C4244: 'initializing': conversion from 'ssize_t' to 'int',
possible loss of data
```

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Tom1 on Sun, 02 Jul 2017 17:49:54 GMT
View Forum Message <> Reply to Message

Hi Oblivion,

I tried SshAsyncRequests example. It compiled nicely on MSC14 32 bit with just a few obvious warnings from libssh2:

```
----- SSH ( MSC14 MSC WIN32 )
Ssh.cpp
SFtp.cpp
Channel.cpp
Scp.cpp
Exec.cpp
Malloc.cpp
libssh2upp.c
c:\upp-10804\bazaar\ssh\libssh2/agent.c(300): warning C4996: 'sprintf': This function or variable may be unsafe. Consider using sprintf_s instead. To disable de
    precation, use _CRT_SECURE_NO_WARNINGS. See online help for details.
c:/program files/windows kits/10/include/10.0.14393.0/ucrt\stdio.h(1769): note: see declaration of 'sprintf'
c:\upp-10804\bazaar\ssh\libssh2/channel.c(1157): warning C4996: 'sprintf': This function or variable may be unsafe. Consider using sprintf_s instead. To disable
     deprecation, use _CRT_SECURE_NO_WARNINGS. See online help for details.
c:/program files/windows kits/10/include/10.0.14393.0/ucrt\stdio.h(1769): note: see declaration of 'sprintf'
c:\upp-10804\bazaar\ssh\libssh2/knownhost.c(957): warning C4996: 'fopen': This function or variable may be unsafe. Consider using fopen_s instead. To disable de
    precation, use _CRT_SECURE_NO_WARNINGS. See online help for details.
c:/program files/windows kits/10/include/10.0.14393.0/ucrt\stdio.h(207): note: see declaration of 'fopen'
c:\upp-10804\bazaar\ssh\libssh2/knownhost.c(1181): warning C4996: 'fopen': This function or variable may be unsafe. Consider using fopen_s instead. To disable d
    eprecation, use _CRT_SECURE_NO_WARNINGS. See online help for details.
c:/program files/windows kits/10/include/10.0.14393.0/ucrt\stdio.h(207): note: see declaration of 'fopen'
c:\upp-10804\bazaar\ssh\libssh2/pem.c(257): warning C4018: '<=': signed/unsigned mismatch
c:\upp-10804\bazaar\ssh\libssh2/userauth.c(541): warning C4996: 'fopen': This function or variable may be unsafe. Consider using fopen_s instead. To disable dep
    recation, use _CRT_SECURE_NO_WARNINGS. See online help for details.
c:/program files/windows kits/10/include/10.0.14393.0/ucrt\stdio.h(207): note: see declaration of 'fopen'
Creating library...
SSH: 7 file(s) built in (0:13.20), 1886 msecs / file, duration = 13547 msecs, parallelization 95%
C:/upp-10804/out/examples-bazaar/SSH/MSC14.\SSH.lib (1624178 B) created in (0:01.96)

OK. (0:13.64)
```

Then it executed with the following results:

Connecting to Rebex public sftp test server...
Unable to request SFTP subsystem
drwx------ 2 demo users 0 Oct 27 2015 .
drwx------ 2 demo users 0 Oct 27 2015 ..
-rw------- 1 demo users 14280 Mar 19 2007 ConsoleClient.png
-rw------- 1 demo users 15091 Mar 19 2007 ConsoleClientSmall.png
-rw------- 1 demo users 15836 Mar 19 2007 FtpDownloader.png
-rw------- 1 demo users 19156 Feb 16 2007 imap-console-client.png
-rw------- 1 demo users 36672 Mar 19 2007 KeyGenerator.png
-rw------- 1 demo users 24029 Mar 19 2007 KeyGeneratorSmall.png
-rw------- 1 demo users 16471 Feb 16 2007 mail-editor.png
-rw------- 1 demo users 35414 Feb 16 2007 mail-send-winforms.png
-rw------- 1 demo users 49011 Feb 16 2007 mime-explorer.png
-rw------- 1 demo users 58024 Mar 19 2007 pocketftp.png
-rw------- 1 demo users 20197 Mar 19 2007 pocketftpSmall.png
-rw------- 1 demo users 20472 Feb 16 2007 pop3-browser.png
-rw------- 1 demo users 11205 Feb 16 2007 pop3-console-client.png
-rw------- 1 demo users 407 Mar 23 2007 readme.txt
-rw------- 1 demo users 11546 Mar 19 2007 ResumableTransfer.png
-rw------- 1 demo users 2635 Mar 19 2007 winceclient.png
-rw------- 1 demo users 6146 Mar 19 2007 winceclientSmall.png
-rw------- 1 demo users 80000 Mar 19 2007 WinFormClient.png
-rw------- 1 demo users 17911 Mar 19 2007 WinFormClientSmall.png
Remote command processed. Return code is 0.
<--- Finished in (0:01.79), exitcode: 0 --->

I wonder if "Unable to request SFTP subsystem" was to be expected here.

Anyway, I will do more testing tomorrow. :)

Thanks and best regards,

Tom

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Sun, 02 Jul 2017 19:14:59 GMT
View Forum Message <> Reply to Message

Hello Koldo and Tom,

Thank you both for testing the code, an dproviding me the crucial feedback!

Quote:- SFtp::Init(), Stop(), and Ssh.WhenWait() do not exist.

Yes, they are removed in favour of a better approach.

I believe initialization and deinitializaton should be made automatic, analogous to RAII.
If you request a resource, say a Scp channel, which should be allocated per request anyway, you simply spawn it and bind it to an ssh session.
It will be automatically deallocated when the object is destroyed, as is with any C++ object:

```
//
Scp scp1;
scp1.Session(session);                    // Scp channel is "scheduled" to be initialized. There is no need for manual deinitialization. It will be deinitialized at the end of the code block.
//Or
Scp scp2(session);                        // Scp channel is "scheduled" to be initialized. There is no need for manual deinitialization. It will be deinitialized at the end of the code block.
```

In the same vein, you don't actually need to explicitly disconnect from the ssh server. (Of course you can, but it is by no means necessary).
Ssh objects automatically attempt to disconnect when they are being destroyed.

WhenWait was a member of JobQueue. But there is no need for WhenWait anymore.
As you can see in the SshAsyncRequests example that I provided with the package, you can use SocketWaitEvent where it is useful.
If you want to include it in a synchronous call, you can override JobQueue::Execute(). It is virtual. :)

As for the old Scp Demo.
Getters and Putters are now in U++ style.

```
 bool Get(Stream& out, const String& path, Gate<int64, int64> progress = Null);
 bool Put(Stream& in, const String& path, long mode, Gate<int64, int64> progress = Null);
```

So the code needs a slight change:

```
Cout() << "Getting file Demo.mo\n";
FileOut fout(AppendFileName(GetDesktopFolder(), "compile_run.sh"));
const char *path = "compile_run.sh";
Scp scp(ssh);
if(!scp.Get(fout, path, [=](int64 total, int64 done) { return false; })) // <--
    Cout() << scp.GetErrorDesc();
fout.Close();
```

Regarding the warnings, I'll try to fix them. Main problem is that Stream return int64 for file size, etc. but libssh2 functions expect size_t.

Hello Tom,

Those warnings are typical and can be suppressed. Usually they are not harmful. But I'll see what I can do.

Quote:I wonder if "Unable to request SFTP subsystem" was to be expected here.

Yes, this is expected. It can happen time to time. But it should only be a rare case. If not, then there might be a problem with initialization.
I'll look into it.

There are two things to note here though:
1) Remote servers are not always reliable. Rebex is fine but response times of it's public test server are somewhat slow.

2) There is a "little" problem with ssh protocol and it's implementations: Ssh session works over a single socket.
Each ssh subsystem, be it sftp or scp, are actually channels on a single socket. And AFAIK there is no official way to "wait" on channels.
So, spawning more than one channel, especially when they are of different subsystems, can result in a such error.
It is also discussed in the libssh2 documents, There is a suggested solution for this by the libssh2 devs, but it is not implemented yet:

libssh2/TODO, lines 80 to 120 reads:

New Transport API
==========

THE PROBLEM

The problem in a nutshell is that when an application opens up multiple channels over a single session, those are all using the same socket. If the application is then using select() to wait for traffic (like any sensible app does) and wants to act on the data when select() tells there is something to for example read, what does an application do?

With our current API, you have to loop over all the channels and read from them to see if they have data. This effectively makes blocking reads impossible. If the app has many channels in a setup like this, it even becomes slow. (The original API had the libssh2_poll_channel_read() and libssh2_poll() to somewhat overcome this hurdle, but they too have pretty much the same problems plus a few others.)

Traffic in the other direction is similarly limited: the app has to try sending to all channels, even though some of them may very well not accept any data at that point.

A SOLUTION

I suggest we introduce two new helper functions:

libssh2_transport_read()

- Read "a bunch" of data from the given socket and returns information to the app about what channels that are now readable (ie they will not block when read from). The function can be called over and over and it will repeatedly return info about what channels that are readable at that moment.

libssh2_transport_write()

- Returns information about what channels that are writable, in the sense that they have windows set from the remote side that allows data to get sent. Writing to one of those channels will not block. Of course, the underlying socket may only accept a certain amount of data, so at the first short return, nothing more should be attempted to get sent until select() (or equivalent) has been used on the master socket again.

I haven't yet figured out a sensible API for how these functions should return that info, but if we agree on the general principles I guess we can work that out.

For the time being, best workaround for this problem seems to be increasing the socket "wait" time and adopting a "max. retries" approach. :)

I am in the process of writing the api docs, and a guide for SSH protocol and package, I'll discuss these problems there too.

Best regards,

Oblivion

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Mon, 03 Jul 2017 05:55:53 GMT
View Forum Message <> Reply to Message

There is a problem that needs to be solved though. I've encountered this error recently.

Connections to servers on localhost (127.0.0.1) fail for some unknown (yet) reason. It needs further investigation.
As far as I can see, this failure doesn't happen when linking against externally (using Cmake)

compiled libssh2.
Probably I am missing some configuration switch (or a combination of switches?).
libssh2 is a strange beast with a configure script which has over 20K LoC. It'll take some time (a week or so) to figure out what I'm missing.
But of course in the meantime I'm open to suggestions, and I accept patches, etc.

Best regards,
Oblivion.

Subject: Re: SFTP or full SSH2 support for U++?
Posted by koldo on Mon, 03 Jul 2017 06:39:21 GMT
View Forum Message <> Reply to Message

Thank you Oblivion. New scp interface works well :)

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Tom1 on Mon, 03 Jul 2017 08:33:14 GMT
View Forum Message <> Reply to Message

Hi Oblivion,

I have successfully tested your examples on Windows 10 Professional 64-bit platform using MSC14, MSC14x64, MSC15 and MSC15x64 build methods.

I guess the usual warnings on the deprecation of functions used by libssh2 could probably be suppressed by using a compiler flag associated to the SSH package. This is just a cosmetic issue though.

Thank you very much for your effort on this task.

Best regards,

Tom

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Mon, 03 Jul 2017 19:51:06 GMT
View Forum Message <> Reply to Message

Hello Koldo and Tom,

It is nice to hear that the package is working for you. :)
I have found the source of the problem that I mentioned in my last message (regarding the error

where the ssh connection to servers on localhost was failing.)
As it turns out, my suspicion was right. A preprocessor directive was missing:
LIBSSH2_DH_GEX_NEW
This preprocessor directive enables the newer diffie-hellman-group-exchange-sha1 syntax, which libssh2 relies on when compiled against recent versions of openssl.

Also I fixed a critical error where the JobQueue can stuck in an infinite loop when the global timeout is set. Therefore please don't forget to update JobQueue too.

You can find the updated package below.

Now that the most annoying problem is solved, I can focus on completing the package (by adding knownhosts, X11 and terminal support, fixing warnings, etc.).

Once the package is complete I am going to open a bazaar topic for SSH package.
And if at that point Mirek and other U++ developers decide that SSH package can be official part of Upp, it has a new BSD license, and permission is hereby explicitly granted.
(I'll gladly continue to maintain the package, since I'll be using it.)

For the time being I'll provide a SVN link to the package, where you can always get the latest version:

> https://sourceforge.net/p/ultimatecomponents/svn/HEAD/tree/t runk/

Best regards,
Oblivion

File Attachments
1) SSH Package (technical preview).zip, downloaded 348 times

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Tom1 on Tue, 04 Jul 2017 06:57:29 GMT
View Forum Message <> Reply to Message

Hi Oblivion,

For some reason the latest SSH.h had the include files for libssh2 changed from:

#include "libssh2/libssh2.h"
#include "libssh2/libssh2_sftp.h"
#include "libssh2/libssh2_publickey.h"

To:

```
#include "libssh2.h"
#include "libssh2_sftp.h"
#include "libssh2_publickey.h"
```

This effectively prevented compilation. I changed it back and now it compiles and runs again OK.

Best regards,

Tom

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Tue, 04 Jul 2017 07:02:42 GMT
View Forum Message <> Reply to Message

Ah, while I was inspecting the cause for the connection error I changed the headers to test some cases.
I was in a hurry, sorry for the inconvenience. :blush:
I'll fix them tonight. Thanks!

Best regards,
Oblivion

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Tom1 on Tue, 04 Jul 2017 07:12:48 GMT
View Forum Message <> Reply to Message

No worries, it was just a minor detail.

Best regards,

Tom

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Sat, 12 Aug 2017 20:49:18 GMT
View Forum Message <> Reply to Message

Hello everyone,

Next update to SSH package technical preview is finally here.

Aside from bug fixes and cleanups, SSH package has now gained full support for three authentication methods (password, publickey, keyboard-interactive) properly.

The crucial part was keyboard interaction (challenge/response) authentication method. It is implemented using a Function.
Also, it is now possible to select between preferred transport methods.
Last but not least, API reference docs are added for the most crucial classes. :)
I am also writing a guide for its usage. But it's not finished yet.

Agents, X11 support, and forwarding support are on their way way too!

Now I'd like to ask you a question. This covers both SSH package and my FTP package.
I'm planning to add to both package network proxy (HTTP/SOCKS4/4a/5) support using the new version of my NetworkProxy package (it will be renamed to NetProxy).

I can think of three strategies:

1) Adding proxy support as an integral part to SSH and FTP packages. This will cause SSH and FTP packages to require NetProxy and I don't really prefer this.
2) Using a compiler flag for enabling proxy support.
3) Since NetworkProxy (and upcoming NetProxy) is a delegate class that simply takes an already initialized TcpSocket, connects it, and hands it over back to user, I can add a callback, say WhenProxy, to Ssh and Ftp classes, and hand over the socket to NetProxy there, and when the connection is established we can proceed. This way both packages won't require NetworkProxy at all, it will be completely optional.

What do you think? Which strategy would be better?

I appreciate suggections, bug reports, criticism, etc.

Here's a digest:

2017-08-11: Initial api reference docs for Ssh, Ssh::SubSystem, SFtp, Scp, Exec, Knownhosts, are added.
2017-07-31: It is now possible to query, get, and set the possible transportation methods and exchange
algorithms. Added Ssh::Method(), Ssh::Methods(), Ssh::GetMethod(), Ssh::GetMethods() methods.
Ssh::Host class is from now on KnownHosts class.
2017-07-21: Authentication methods (password, public key, keyboard interaction) are properly implemented.
From now on it is possible to choose between authentication methods both on initialization,
and on-the-fly (i.e. while logging in, using WhenAuth callback).
KnownHosts class is added to the package. This class provides basic known hosts support.
It is now possible to verify and trust servers.

Best regards,

Oblivion

File Attachments
1) SSH Package (technical preview)-2017811.zip, downloaded 297 times

---

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Tom1 on Mon, 14 Aug 2017 10:41:27 GMT
View Forum Message <> Reply to Message

Hi Oblivion,

Thanks for the update. :)

Not sure if this has anything to do with the update, but I have trouble concatenating the following procedure to conditionally create a directory:

```
...
 if(session.IsSuccess()){
  SFtp::DirList list;
  SFtp sftp(session);

  sftp.Clear();
  sftp.StartListDir(directoryname,list);
  sftp.StartClose();
  if(sftp.Execute()) return true; // Reading the directory successfully proves it exists

  sftp.Clear();
  sftp.StartMakeDir(directoryname,0755);
  sftp.StartClose();
  return sftp.Execute();
 }
...
```

The MakeDir part fails when done after ListDir. If I establish a separate SFtp instance for both ListDir and MakeDir, it works OK. Is it supposed to work that way?

Best regards,

Tom

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Mon, 14 Aug 2017 12:26:06 GMT
View Forum Message <> Reply to Message

Hello Tom,

tl;dr: I'll write a better reply and explanation, once I get back home. Also I'm writing a guide on it's usage. :)

I can see what you are trying to achieve, but that's kinda' against the whole concept.
I mean, in the snippet you provided, you are emulating synchronous calls with asynchronous methods, which is basically legitimate, but already done in the SSH packages synchronous calls :) :

Synchrounous version may be (I don't know the code-flow):

```
if(session.IsSuccess()){
  SFtp::DirList list;
  SFtp sftp(session);
  if(!sftp.ListDir((directoryname, list))
                return sftp.MakeDir(directoryname, 0755);

}
```

Asynchronous version (which really shouldn't be written this way. Just do not use this. Only to give you the idea.) :

```
if(session.IsSuccess()){
  SFtp::DirList list;
  SFtp sftp(session);

                sftp.StartListDir(directoryname, list);
                while(sftp.Do());
                if(sftp.IsFailure()) {
                    sftp.StartMakeDir(directoryname, 0755);
                    while(sftp.Do());
                    return sftp.IsSucces();
                }
}
```

You dont' need to call Clear() at all. Queue will be cleared automatically.
It is supplied as a safety measure (which I'll cover it in the guide) for creating (custom) synchronous methods by the user.

By the way, I am going to add high-level methods, such as IsFileExists(), IsDirectoryExists(), etc. :)

Thanks for the feedback!

Best regards,
Oblivion

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Tom1 on Mon, 14 Aug 2017 12:46:53 GMT
View Forum Message <> Reply to Message

Hi,

Thanks for your prompt reply. (Although I'm not in any hurry with this, whatsoever.)

Well, I tried the async approach just to test, but I had the exact same problem with the synchronous one too .. i.e. This fails:

```
if(session.IsSuccess()){
  SFtp::DirList list;
  SFtp sftp(session);
  if(!sftp.ListDir(directoryname, list))
                return sftp.MakeDir(directoryname, 0755);

}
```

But this works:

```
if(session.IsSuccess()){
  SFtp::DirList list;
  SFtp sftp(session);
  if(!sftp.ListDir(directoryname, list)){
   SFtp sftp2(session);
   return sftp2.MakeDir(directoryname, 0755);
  }

}
```

The high level helpers, like you pictured in your message, would be great. Please check the naming of U++ counterparts to keep the access easy. (BTW: I think that I have seen IsOK() quite frequently in U++ instead of IsSuccess(), but I'm not sure if these are supposed to mean exactly the same thing.)

Thanks and best regards,

Tom

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Mon, 14 Aug 2017 12:52:24 GMT

Hello Tom,

Quote:(BTW: I think that I have seen IsOK() quite frequently in U++ instead of IsSuccess(), but I'm not sure if these are supposed to mean exactly the same thing.)

Queue model is an abstraction of Upp::HttpRequest's async model. I took Do(), IsSuccess(), IsFailure(), InProgress() from there. I think they are legitimate.

And I see the source of the problem now, I'll fix it asap.
Thanks!

Best regards,
Oblivion

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Tom1 on Mon, 14 Aug 2017 12:57:13 GMT

Hi,

I never used HttpRequest, so I'm not familiar with this.

But no worries; I trust you know best what you're up to! :)

Best regards,

Tom

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Sun, 15 Oct 2017 21:10:37 GMT

Hello Mirek, Tom, Koldo, and everyone,

It's been a while since I last updated the package.

SSH package's been through a major rewrite, and I wanted to have a solid foundation to build it upon. Well, I believe it is finally here.

New design hopefullt fixes the main problems of the previous one. To do this I focused on the major component of SSH package: SFtp, since it was the most complex part of it.

I am here to share the results. Below package contains only Ssh and SFtp components for the time being. But please take your time to examine and test SFtp. It can do some

nice tricks easily, such as peeking and poking at files, or -experimental- multithreading support (we'll return to that later.).

I also included a simple testcase where you can see how SFtp class behaves. Note that, every other component (be it SCP, channel, exec) will behave in a similar way. :)

Here is the highlights, and changes:

1) This version brings a major redesign of overall components. SSH package's classes are redesigned around a core class named SshCore. SshCore allows a uniform interface for both blocking and non-blocking modes of operation. Visible result of this new design is a single set of methods for both modes of operation for each class. Methods starting with "Start" prefix are removed in favor of a simple "NonBlocking()" switch.

2) In accordance with this behaviour, a sort of "feature parity" is successfully kept, thanks to the new design. Namely, in non-blocking mode, any result can be gathered using the new GetResult() method, which returns Value.

3) All SSH package classes now has pick semantics.

4) New design allows creation of higher-level ("complex") SFTP methods where needed. Unlike the simple SFTP methods, which work on file handles, Complex methods take care of (allocate/free) file handles internally.

5) Experimental multithreading support is added (using AsyncWork). This is still at a primitive stage.

6) Network proxy support is added. This new feature uses NetProxy (Http/Socks4/4a/5) package It works through a plugin, provided via Ssh::WhenProxy callback. And it is completely optional.

7) Cancellation mechanism is added. Any operation can be cancelled at any time using the Cancel() method.

8) SFtp::DirEntry class now has a ToString() method. (It will also gain an ToXml() method.) This method will give an output similar to Unix ls command.

9) libssh2 configured to use OpenSSL by default (WinCNG will be made a compile-time option in the next release).

The code is not entirely cleaned-up, so there maybe also small cosmetic issues (for now.)

I appreciate feedbacks, patches, criticism, etc.

[See below posts for updated package]

Cheers!

Oblivion

Subject: Re: SFTP or full SSH2 support for U++?
Posted by koldo on Mon, 16 Oct 2017 06:58:45 GMT
View Forum Message <> Reply to Message

Thank you :)

Subject: Re: SSH2 wrapper for U++
Posted by Oblivion on Wed, 18 Oct 2017 22:55:27 GMT
View Forum Message <> Reply to Message

Hello,

SSH package is updated.

2017-10-17: Class names finalized:
Core/base class -> Ssh
Session class -> SshSession
SFtp class -> SFtp
Channel class -> SshChannel
Scp class -> Scp
Exec class -> SshExec
Knownhosts class -> SshHosts
SshAgents -> (will be added as such)

CreateSFtp(), CreateExec(), CreateScp(), CreateChannel() methods added to SshSession
class.
Logging further refined.
Error management further refined.

New version of crucial SSH components are re-added to the packaet.

As you can see in the above history entry, component names are being finalized. I am satisfied with these new namings, and would like to hear your commennts.

Also I added CreateSFtp(), CreateExec(), CreateScp(), CreateChannel() methods to SshSession class.

Since now we have pick semantics, these methods are IMO more elegant than explicity constructing subsystem objecs.

A simple use case would be as follows (below code is a part of SFtpDemo, by the way. (since Rebex test server is actually a SFtp server, with exec support, I did not change the example name):

```
void GetFileInfoViaExec(SshSession& session)
{
 DLOG("---- Getting directory listing of " << file << " using SshExec (blocking):");
 auto exec = session.CreateExec();
 DDUMP(exec(String(cmd) + file, Cout(), Cerr()));

}
```

As I announced in my previous message, SSH package recently gained a completely optional NetProxy support.
(see Bazaar section https:// www.ultimatepp.org/forums/index.php?t=msg&th=10132&goto=48812&#msg_48812 for NetProxy package)
Here is the minimal example code (this is a real test code that works):

```
CONSOLE_APP_MAIN
{
    StdLogSetup(LOG_FILE | LOG_COUT | LOG_TIMESTAMP);

//  Ssh::Trace();
    SshSession session;
    session.WhenProxy = [=](TcpSocket& sock) {
        NetProxy proxy(sock, proxy_host, proxy_port);
        return proxy.Timeout(10000)
                    .Socks5()
                    .Auth(proxy_username, proxy_password)
                    .Connect(ssh_host, ssh_port);
    };
    if(session.Connect(ssh_host, ssh_port, username, password)) {
        auto exec = session.CreateExec();
        DDUMP(exec("ls -l /readme.txt", Cout(), Cerr());
    }
    else Cerr() << session.GetErrorDesc() << "\n";
}
```

RTTI support is also re-implemented.

Soon I'll move SSH package out of technical preview phase to beta. But before that I have to add documentation, a proper multithreading support, and missing classes (SshHost, SshAgents).

On multithreading:

The design of SSH core (base) class "ssh" allowed me to easily add an "internal" MT support using a compile-time switch (I did not yet publish that code).
But I haven't really decided to add MT that way. Using convenience functions (SFtpGet, SFtpPut, ScpGet, ScpPut, AsyncExec) seem a simpler and better managable way to go.
But I'd like to hear yout thoughts on that matter too.

I appreciate reviews, bug reports, patches, suggestions, criticism.

Best regards,
Oblivion.

File Attachments

1) SSH-SecondIteration.zip, downloaded 279 times

Subject: Re: SSH2 wrapper for U++
Posted by Tom1 on Thu, 19 Oct 2017 12:16:32 GMT
View Forum Message <> Reply to Message

Hi Oblivion,

I downloaded this latest version now after using the version from a few months back.

In order to successfully compile the code, I had to comment out the entire contents of SFtpMT.cpp and the following lines in SFtp.h:

```
AsyncWork<String>   SFtpGet(SshSession& session, const String& path, Gate<int64, int64> progress = Null);
AsyncWork<void>     SFtpGet(SshSession& session, const String& source, const String& target, Gate<int64, int64> progress = Null);
```

Additionally, I had to comment out the following in Core.cpp:

DLOG("Initializing libssh2...");
and
DLOG("Deinitializing libssh2...");

(There was some 0x40 character embedded in those lines that MSC did not like.)

-

Then there's one question: Now that the names have changed, is there a replacement for session.IsSuccess() ? (I just removed these calls in my code to see if SFTP still works. It did :))

Thanks again for your work on SSH!

Best regards,

Tom

Subject: Re: SSH2 wrapper for U++
Posted by Oblivion on Thu, 19 Oct 2017 13:09:25 GMT
View Forum Message <> Reply to Message

Hello Tom,

Quote:I downloaded this latest version now after using the version from a few months back.

In order to successfully compile the code, I had to comment out the entire contents of SFtpMT.cpp and the following lines in SFtp.h:

AsyncWork<String> SFtpGet(SshSession& session, const String& path, Gate<int64, int64> progress = Null);
AsyncWork<void> SFtpGet(SshSession& session, const String& source, const String& target, Gate<int64, int64> progress = Null);

Ah yes, that's because I use U++ trunk (latest nightly builds).
In case you didn't notice, there is now a class called AsyncWork in U++ core, which is a MT helper. Those functions use that. :)

Quote:

Then there's one question: Now that the names have changed, is there a replacement for session.IsSuccess() ? (I just removed these calls in my code to see if SFTP still works. It did

Smile)

You don't need it. I got rid of the old ones. There is only IsError(). :)

You can use the IsError() method to check success when it is necessary.

Did you look into the provided SFtpDemo app? Take a look into it. It demonstartes the basic usage pattern for SFtp (same pattern is valid for other components too)

If you use blocking mode, then you can simply check the return code (IsError will be still available though.)

In non-blocking mode, A successful operation means there is no error. So again, you can simply check IsError(), and then get the result with GetResult() method if the invoked method returns a value.

By the way, GetResult() method is needed in non-blocking mode for such methods as FileExists(), for we need to separate the state of operation (failure/success) from the return value of operation.

Non-blocking:

```
void GetFileSize(SshSession& session)
{
 DLOG("---- Getting file size of " << file << " (non-blocking):");
 auto sftp = session.CreateSFtp();
 sftp.NonBlocking().GetSize(file);
 while(sftp.Do())
  Sleep(1);
 if(sftp.IsError())
  DDUMP(sftp.GetErrorDesc());
 else
  DDUMP((int64) sftp.GetResult()); // In non-blocking mode, results can be
       // "harvested" using GetResult() method.
}

void FileExists(SshSession& session)
{
 DLOG("---- Test if " << file << " exists (non-blocking):");
 auto sftp = session.CreateSFtp();
 sftp.NonBlocking().FileExists(file);
 while(sftp.Do())
  Sleep(1);
 if(sftp.IsError())
  DDUMP(sftp.GetErrorDesc());
 else
```

```
  DDUMP((bool) sftp.GetResult());

        // Alternatively (blocking)
        if(sftp.NonBlocking(false).FileExists(file))
          Cout() << file << " exists.\n";
        else
          Cerr() << sftp.GetErrorDesc() << '\n';
}
.
```

Blocking:

```
 DLOG("---- Downloading file " << file << " directly to Cout() (blocking):");
 auto sftp = session.CreateSFtp();
 if(!sftp.Get(file, Cout()))
  DDUMP(sftp.GetErrorDesc());
 else
  DLOG("Done.");
```

Or

```
void GetDirectoryList(SshSession& session)
{
 DLOG("---- Getting directory listing of " << dir << " (blocking):");
 auto sftp = session.CreateSFtp();
 SFtp::DirList list;
 if(sftp.ListDir(dir, list)) {
  for(auto& e : list)
   DDUMP(e);
 }
 else DDUMP(sftp.GetErrorDesc());
}
```

I'll explain the details in documentation, but it is really very simple.

Note that I "removed" also the batch processing mode. This new version of SSH package can only process one request at a time. (well, from the user POV, at least. Basically it still uses a queue, but only internally, and in a well-determined way)

Best regards,
Oblivion

Subject: Re: SSH2 wrapper for U++
Posted by Tom1 on Mon, 23 Oct 2017 06:29:10 GMT
View Forum Message <> Reply to Message

Hi Oblivion,

Thanks for explaining the new details. I did not read the SFtpDemo this time, since I had already implemented client calls in my own software. So in effect I was just updating the SFTP code to its latest version. I should have paid more attention to the changes.

Anyway, thanks for the update! I'll let you know if I run into any trouble with it.

Best regards,

Tom

Subject: SSH package for U++ (alpha version)
Posted by Oblivion on Mon, 13 Nov 2017 20:25:12 GMT
View Forum Message <> Reply to Message

Hello,

SSH package is updated.
This version is the first "alpha" release.

This version brings,

- Multithreading support for SFtp, Scp, and SshExec classes.

- Ssh agent authentication support.

- SocketWaitEvents support for non-blocking I/O

- Ssh known hosts support (re-added).

- API documentation (updated)

- SFtp::DirEntry class gained ToXml() method.

- Keyboard (challenge/response) authentication (re-added)

- Updated libssh2

- Various bugfixes

I will soon add SshShell support. In fact there is already a prototype of SshShell for Posix based OSes. It can execute remote commands in real-time, and allows using editors such as nano, vim, emacs on console. It also works with GUI. :) I will add it once I clean its source code.

However, adding shell support on windows is somewhat tricky (at least, adding a console (CLI) based version is not so trivial, for standard I/O handling is different on windows. I need to find a reliable way before I can include it in the pacakge, so it'll take some time...)

I set up a GIT repo some time ago, where you can find the latest versions of my public U++ packages:

https://github.com/ismail-yilmaz/upp-components/tree/master/ Core/SSH

Below you can find the latest package. (I will open a Bazaar topic from the next release on, and update the package regularly.)

It is tested on Linux and Windows (GCC/MINGW 7.2, MSC 14 (2017)).

Suggestions, criticism, bug reports, reviews, testing is always welcome,

Cheers!

Oblivion

File Attachments

1) SSH - 20171112a.zip, downloaded 335 times

Subject: Re: SFTP or full SSH2 support for U++?
Posted by alkema_jm on Tue, 14 Nov 2017 07:03:36 GMT
View Forum Message <> Reply to Message

LS,

I use upp-win-11459.7z, vs2017 and Windows10 , I downloaded https://svn.code.sf.net/p/ultimatecomponents/svn/, I put the trunk directories in the bazaar directory.

IpAddrInfo addrinfo;

Error is that the variable 'addrinfo' is a type/structure in Ultimate. To solve error, is renaming variable 'addrinfo' in something else for example 'addrinfo2'.

File Attachments
1) clientsockets_error1.jpg, downloaded 647 times

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Tue, 14 Nov 2017 07:38:32 GMT
View Forum Message <> Reply to Message

Quote:
alkema_jm
index.php?t=getfile&id=5432&private=0LS,

I use upp-win-11459.7z, vs2017 and Windows10 , I downloaded https://svn.code.sf.net/p/ultimatecomponents/svn/, I put the trunk directories in the bazaar directory.

IpAddrInfo addrinfo;

Error is that the variable 'addrinfo' is a type/structure in Ultimate. To solve error, is renaming variable 'addrinfo' in something else for example 'addrinfo2'.

Hello alkema_jm,

Thank you for notifiying me about the issue.

SVN repo is not yet updated. SSH package in SVN repo is outdated. I will update it tonight.

https://github.com/ismail-yilmaz/upp-components/tree/master/ Core/SSH

However, the name clash you've mentioned is already fixed in the GIT version above.

Note that the new version is not entirely compatible with the old one. I suggest you update to it. It is a re-write with a much better design with a ton of bug-fixes and new features. (Transition should be fairly easy though),

Ps.My previous message also includes the latest zip file which contains an example code demonstrating the basic usage of the latest SSH package. You may want to chech that too.

Best regards,
Oblivion

Subject: Re: SFTP or full SSH2 support for U++?

Posted by alkema_jm on Tue, 14 Nov 2017 10:57:05 GMT
View Forum Message <> Reply to Message

Hello Oblivion,

I put the code in my C:\dev\upp\bazaar directory. I use each time fresh upp-win-11459.7z (source files), vs2017 and Windows10 See for the results the screendumps below.

Greetings Jan Marco

File Attachments
1) sftp_compile_screendumps.jpg, downloaded 614 times

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Tue, 14 Nov 2017 11:41:05 GMT
View Forum Message <> Reply to Message

Hello Jan Marco,

Thank you very much for feedback. :)
Warnings aside (they are not really important here and will be addressed soon),

Quick fix for SFtpDemo: You can make SFtpDemo work by changing SFtpGet to SFtp::AsyncGet (in SFtpDemo.cpp, at line 83)

Other error messages: SSH, Job, NetProxy and FTP packages are 3rd party U++ libraries, and they are meant to be used in applications or other libraries.
That's why you are getting linker errors about _main.

Best regards,
Oblivion

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Tom1 on Tue, 14 Nov 2017 13:27:38 GMT
View Forum Message <> Reply to Message

Hi Oblivion,

Thanks for the update. I just downloaded it and compiled it in and it still works just fine within my app. (That is for the SFTP part I'm using :)

Thanks and best regards,

Tom

Subject: Re: SFTP or full SSH2 support for U++?
Posted by Oblivion on Tue, 14 Nov 2017 20:51:26 GMT
View Forum Message <> Reply to Message

Hello Tom,

I am glad that it works for you. :)

Since SSH package has now passed the technical-preview stage, and has a mostly stable API, I am opening a new topic on the Bazaar section and officially announcing it as a SSH supplement for Ultimate++ core library.

From now on you can always find latest version, help, and SSH package-related news there:

https://www.ultimatepp.org/forums/index.php?t=msg&th=101 72&start=0&

Best regards,
Oblivion