Subject: SSH package for U++
Posted by Oblivion on Tue, 14 Nov 2017 20:59:37 GMT
View Forum Message <> Reply to Message

Hello guys,

I am officially announcing the release of "beta" version of SSH package.
Package is tested on Arch Linux (Kernel: 4.13, GCC 7.2) and Windows (7/10, MinGW 7.2/MSC 14 (2017))

SSH package for U++

SSH package is a feautre-rich, flexible yet very easy to use libssh2 wrapper for Ultimate++.
It supports both console and GUI-based applications on POSIX-compliant operating systems and MS Windows (tm).

Currently it is in beta (version 1) stage.

Classes:

- Base (core) class -> Ssh
  - Ssh session -----> SshSession
  - Sftp subsystem -----> SFtp
  - Ssh channel -----> SshChannel
    - Scp channel -----> Scp
    - Exec channel -----> SshExec
    - Real-time interactive shell -----> SshShell
    - X11 forwarding -----> SshShell (as operation mode)
    - Tcp/IP and port forwarding -----> SshTunnel

- Known hosts manager -> SshHosts

Features and Highlights:

- Ssh-derived classes have pick semantics, based on RAII principle, support RTTI, and allow polymorphism (i.e. different classes can be stored in the same arrays, etc.) through a common interface.
- Uses U++ memory allocators (Native allocators is also a compile-time option)
- Uses OpenSSL by default.
- Supports time-constrained, blocking and non-blocking operation modes.
- Supoorts multithreaded file transfers and remote command execution (exec), using worker threads.
- Supports 3rd-party network proxies. (Such as NetProxy)
- Supports known hosts verification mechanism.
- Supports password, public key, host-based, and keyboard-interactive authentication methods.

- Supports ssh-agents.
- Supports real-time interactive command line (shell) interface with both console and GUI integration (works on Windows and Posix-compliant OS'es)
- Supports multiple X11 connection forwarding.
- Supports Tcp/IP and port forwarding.
- Supports detailed (full) debug logging.

Todo:

- Add more high level methods.
- Refactor Ssh (core) class.
- Improve documentation.

Reference examples:

- SFtpGet: Demonstrates basic SFtp file download in blocking mode.
- SFtpGetNB: Demonstrates basic SFtp file download in non-blocking mode.
- SFtpGetMT: Demonstrates basic SFtp file download, using a worker thread.
- SFtpGUI: Demonstrates a basic SFtp browser with GUI (with upload, download, mkdir, rename, delete commands).
- SFtpMultiGetMT: Demonstrates SFtp dir listing and concurrent file transfers, using worker threads.
- SFtpConsumerGet: Demonstrates the usage of a consumer function for SFtp download in blocking mode.
- SFtpConsumerGetMT: Demonstrates the usage of a consumer function for SFtp download, using a worker thread.
- SshExec: Demonstrates basic Ssh remote command execution in blocking mode.
- SshExecNB: Demonstrates basic Ssh remote command execution in non-blocking mode.
- SshExecMT: Demonstrates basic Ssh remote command execution, using a worker thread.
- SshKeyboardAuth: Demonstrates basic Ssh interactive (challenge-response) authentication method in blocking mode.
- SshLoggingExample: Demonstrates the logging capabilities of SSH package.
- SshOverTor: Demonstrates a basic Ssh connection over TOR, using a third-party network proxy adapter.
- SshPolymorphismNB: Demonstrates the polymorphism capability of SSH channels in non-blocking mode.
- SshShell: Demonstrates a basic, real-time SSH shell console in blocking mode.
- SshShellNB: Demonstrates a basic, real-time SSH shell console in non-blocking mode.
- SshShellX11: Demonstrates a basic, real-time SSH shell console with multiple X11 forwarding.
- SshShellGUI: Demonstrates the basic GUI integration of SshShell class with multiple X11 forwarding, in non-blocking mode.
- SshTunnelExample: Demonstrates the basic SSH tunneling (as tunnel/server) in blocking mode.

Below you can find the latest package and the GIT address where you can always get the latest version.

GIT repo: https://github.com/ismail-yilmaz/upp-components/tree/master/ Core/SSH
Examples: https://github.com/ismail-yilmaz/upp-components/tree/master/ Examples
Older Version: https://github.com/ismail-yilmaz/upp-components/tree/master/ Attic/SSH

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

Best regards,
Oblivion

File Attachments

1) SshPackageAndExamples.zip, downloaded 372 times

Subject: Re: SSH package for U++
Posted by Oblivion on Fri, 17 Nov 2017 21:15:05 GMT
View Forum Message <> Reply to Message

Hello,

I am currenty writing a simple SFTP (GUI) browser example, which I will include in the package, and upload it to GIT repo.
In the meantime, for those who wonder how to do non-blocking operations with SFtp, here is a simple example (it is a real example which can be compiled.):

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

using namespace Upp;

CONSOLE_APP_MAIN
{
    // This example demonstrates (in non-blocking mode):
    // 1) Reading the size of a file.
    // 2) Reading the content of a file (into a String).
    // 3) Reading the content of a directory as XML.

    int CMD_GET = 0, CMD_LIST = 0, CMD_SIZE = 0;
    const char *file = "/readme.txt";
    SFtp::DirList ls;

    Ssh::Trace();
    SshSession session;
```

```
    if(session.Timeout(30000).Connect("test.rebex.net", 22, "demo", "password")) {
        Array<SFtp> sftps;
        for(int i = 0; i < 3; i++) {
            auto& sftp = sftps.Add(new SFtp(session));
            sftp.NonBlocking();
            switch(i) {
                case 0: sftp.Get(file);  CMD_GET = sftp.GetId(); break;
                case 1: sftp.GetSize(file);  CMD_SIZE = sftp.GetId(); break;
                case 2: sftp.ListDir("/pub/example/", ls); CMD_LIST = sftp.GetId(); break;
                default: NEVER();
            }
        }
        while(!sftps.IsEmpty()) {
            for(int i = 0 ; i < sftps.GetCount(); i++) {
                SocketWaitEvent we;
                auto& sftp = sftps[i];
                sftp.AddTo(we);
                we.Wait(10);
                if(!sftp.Do()) {
                    if(sftp.IsError())
                        Cerr() << sftp.GetErrorDesc() << '\n';
                    else {
                        if(sftp.GetId() == CMD_GET) {
                            Cout() << sftp.GetResult();
                        }
                        else
                        if(sftp.GetId() == CMD_SIZE) {
                            Cout() << Format("Size of %s is %d bytes\n", file, sftp.GetResult());
                        }
                        else
                        if(sftp.GetId() == CMD_LIST) {
                            for(auto& e : ls)
                                Cout() << e.ToXml() << '\n';
                        }
                    }
                    sftps.Remove(i);
                    break;
                }
            }
        }
    }
    else
        Cerr() << session.GetErrorDesc();

}
```

Best regards,

Oblivion

Subject: Re: SSH package for U++
Posted by Oblivion on Sat, 18 Nov 2017 14:57:01 GMT
View Forum Message <> Reply to Message

Hello,

SSH package is updated.
Package now contains six basic examples.
These examples demonstrate file transfer (get) and remote command execution (exec) in blocking, non-blocking (NB) modes, and using worker threads (MT)

- SFtpGet
- SFtpGetNB
- SFtpGetMT

- SshExec
- SshExecNB
- SshExecMT

You can find the updated package and the address of related GIT repo in the first message of this topic.

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by Oblivion on Sun, 19 Nov 2017 17:21:06 GMT
View Forum Message <> Reply to Message

Hello,

SSH package and examples are updated. This update fixes a critical bug in libssh2, which leads to heap leaks. I strongly recommend the update.

Also SFtp gained two useful methods:

- GetCurrentDir()
- GetParentDir()

Bug reports, patches, criticism, suggestions, or any questions are always welcome.

You can always find the updated package and the address of its GIT repo in the first message of this topic.

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by Oblivion on Mon, 20 Nov 2017 23:38:53 GMT
View Forum Message <> Reply to Message

Hello,

SSH package & examples are updated.
Package gained 2 new examples. One is simple, other is relatively complex.

SshKeyboardAuth: Demonstrates SSH keyboard interactive (challange/response) authentication method.

SFtpMultiGet: Demonstrates download of multiple files simultaneously in SFTP non-blocking mode (non MT).

Bug reports, patches, criticism, suggestions, or any questions are always welcome.

You can always find the updated package and the address of its GIT repo in the first message of this topic.

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by koldo on Thu, 23 Nov 2017 08:07:12 GMT
View Forum Message <> Reply to Message

:)

Subject: Re: SSH package for U++
Posted by Oblivion on Fri, 24 Nov 2017 20:03:21 GMT

Hello,

SSH package is updated.

New method:

bool SshSession::Connect(const String& url)

Connects to a SSH2 server specified by the url. Returns true on success.
Syntax of the URL is as follows:

[ssh|scp|sftp|exec]://[user:password@]host[:port]

In time, the syntax will be improved (optional args will be added.)

Also a new example (SFtpMultiGetMT) is added to the examples.
This example downloads 4 files asynchronously, using worker threads, and is the multithreaded counterpart of SFtpMultiGetNB (non-blocking).
It also demonstrates the capabilities of the U++ AsyncWork worker threads.

Bug reports, patches, criticism, suggestions, or any questions are always welcome.

You can always find the updated package and the address of its GIT repo in the first message of this topic.

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by alkema_jm on Sun, 26 Nov 2017 03:51:32 GMT

Hello Oblivion

>suggestions

Maybe that swish can offer some functionality? N.B. Proper Windows Explorer integration.
Description

Access your remote files over SFTP directly from Windows Explorer and 'My Computer'.
https://sourceforge.net/projects/swish/

Easy SFTP for Windows
Swish adds support for SFTP to Windows Explorer so you can access your files on another computer securely via SSH.

Swish is easy to use because it integrates seamlessly with Windows Explorer so working with remote files feels just like working with the ones on your local computer.

We believe Swish is a better option for typical Windows users than other SFTP clients because it is so easy to use. Download it now and decide for yourself.

http://www.swish-sftp.org/

Features
• SFTP file transfer
• Proper Windows Explorer integration

Greetings Jan Marco

Subject: Re: SSH package for U++
Posted by Oblivion on Sun, 26 Nov 2017 08:59:13 GMT
View Forum Message <> Reply to Message

Hello Jan Marco,

Thank you very much for your comments and suggestions. :)

I've tested swish and it does a decent job for sure.

However, I believe SSH package and swish do not fall under the same categories. Their scope and intended audience seems to be different.

From its website: "Swish is a plugin for Microsoft Windows Explorer that adds support for SFTP."

It is a decent plugin for Windows Explorer, it falls under applications/plugins category, if I may.

Naturally, it is limited to Windows platform, and as far as I can understand, it uses OpenSSL exclusively.

And it seems to be limited to SFTP subsystem.

On the other hand SSH package isn't an app or plugin written by U++. It is an easy to use SSH2 library (it covers full SSH2 ecosystem: Sftp, channels, exec, scp, shell, agents) for U++, and its intended audience is exclusively U++ developers.

SSH uses libssh2 under the hood. libssh2 is a widely used and extensively tested, multiplatform SSH2 library with BSD license. SSH package supports what libssh2 supports.

Therefore SSH package is not limited to windows. In fact, it isn't developed on windows. In theory SSH package can run on what U++ and libssh2 can run on and this includes, but not limited to, Linux and Windows.

Also, while SSH package uses OpenSSL by default, it is also possible to use it with WinCNG and GnuTLS. (Currently through a config file, in the near future via compiler flag.)

In fact there is nothing that prevents a developer to come up with a similar plugin to swish, using SSH pacakge.(Maybe it'll take a lot less time to develop one using SSH package, since the main purpose of SSH package is to let developers focus on other aspects of their apps.)

One major advantage (for commercial app developers) of SSH package over swish is that SSH package (and underlying libssh2) uses BSD license.

To sum up:

1) Swish offers good functionality, but the functionality it offers is mostly on application-level. Hence they should be implemented by the application developers (using SSH package).
2) Writing some example app similar to swish would be platform specific. Therefore I am writing a simple SFtpBrowser example, which will work on any platform that U++ supports. :)

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by alkema_jm on Thu, 30 Nov 2017 22:55:34 GMT
View Forum Message <> Reply to Message

Hello Oblivion,

Today I made a Visio diagram of the different configurations of Swish and FileZilla in relation to SSH-Ultimate++.

Greetings Jan Marco

File Attachments

1) sftp_ultimate_06.gif, downloaded 1375 times

Subject: Re: SSH package for U++
Posted by alkema_jm on Fri, 01 Dec 2017 11:33:16 GMT
View Forum Message <> Reply to Message

Hello Oblivion ,

>Thank you very much for your comments and suggestions.

Thank you that you ask feedback :p

>I've tested swish and it does a decent job for sure.

I tried (/test) to compile Swish. At one point i could start it, but it crasht.

> However, I believe SSH package and swish do not fall under the same categories. Their scope and intended audience seems to be different.

Ok, I think in software components. I like the Windows Explorer integration of Swish only.

>From its website: "Swish is a plugin for Microsoft Windows Explorer that adds support for SFTP."

It is a decent plugin for Windows Explorer, it falls under applications/plugins category, if I may.

Yes, I like Microsoft Windows Explorer integration part of Swish.

>Naturally, it is limited to Windows platform, and as far as I can understand, it uses OpenSSL exclusively.

I don't like. "One size fits all". Some solutions (for example Microsoft Windows Explorer integration) can be better in specific (native) solutions.

>And it seems to be limited to SFTP subsystem.

The FileZilla engine 'uses' more protocols (storj) :

> On the other hand SSH package isn't an app or plugin written by U++. It is an easy to use SSH2 library (it covers full SSH2 ecosystem: Sftp, channels, exec, scp, shell, agents) for U++, and its intended audience is exclusively U++ developers.

I like the SSH2 econsystem very much :p

> SSH uses libssh2 under the hood. libssh2 is a widely used and extensively tested, multiplatform SSH2 library with BSD license. SSH package supports what libssh2 supports.

Ok, You limit SSH package to "libssh2"?

> Therefore SSH package is not limited to windows. In fact, it isn't developed on windows. In theory SSH package can run on what U++ and libssh2 can run on and this includes, but not

limited to, Linux and Windows.

I think it is usefull to think in interfaces (and api) of software components. I separated the (General) FileZilla engine code. I will try to interface with libssh2 (SSH package). I haven't look in the libssh2 code if it is possible.

> Also, while SSH package uses OpenSSL by default, it is also possible to use it with WinCNG and GnuTLS. (Currently through a config file, in the near future via compiler flag.)

FileZilla uses GnuTLS.

> In fact there is nothing that prevents a developer to come up with a similar plugin to swish, using SSH pacakge.(Maybe it'll take a lot less time to develop one using SSH package, since the main purpose of SSH package is to let developers focus on other aspects of their apps.)

I like your SSH package very much, because it Works :p

> One major advantage (for commercial app developers) of SSH package over swish is that SSH package (and underlying libssh2) uses BSD license.

I am not a company. Licenses are not very interesting subject for me.

> To sum up:

>1) Swish offers good functionality, but the functionality it offers is mostly on application-level. Hence they should be implemented by the application developers (using SSH package).

Ok.

2) Writing some example app similar to swish would be platform specific. Therefore I am writing a simple SFtpBrowser example, which will work on any platform that U++ supports.

Ok, a simple example is alway usefull to see that it Works. I think that a more complex example als FileZilla is not difficult to migrate to Ultimate++.

Greetings Jan Marco

File Attachments

1) protocolinfos_engine_filezilla.jpg, downloaded 1412 times

Subject: Re: SSH package for U++
Posted by Oblivion on Fri, 01 Dec 2017 13:35:31 GMT
View Forum Message <> Reply to Message

Hello Jan Marco,

I really appreciate you feedback. Thank you very much!

Quote:The FileZilla engine 'uses' more protocols (storj) :

We already have HTTP(S)(HttpRequest class does that) and FTP(S). Have you looked at the FTP package that I wrote? (https://www.ultimatepp.org/forums/index.php?t=msg&th=8899&goto=43053&#msg_43053)
It covers FTP, and FTPS (FTP over explicit TLS/SSL). Check the video too. It shows what it is capable of: (https://vimeo.com/214530673)

The only protocol we seem to lack is StorJ, which is not really hard to implement. Maybe I should add it to my Todo list. :)

Quote:Ok, a simple example is alway usefull to see that it Works. I think that a more complex example als FileZilla is not difficult to migrate to Ultimate++.

FTP class I mentioned above contains an example called FtpBrowser. That example will eventually evolve into a MultiBrowser (FTP(S)/SFtp/HTTP(S)) after I finalized the SSH package. (In fact it contains a package called BrowserCtrl, which is meant to be a gui-frontend for filesystem-agnostic file browsers).
So while I'm not going to port filezilla to U++ (it'll need a lot of work), I'll come up with a simpler remote file browser demo. :)

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by alkema_jm on Sat, 02 Dec 2017 08:29:15 GMT
View Forum Message <> Reply to Message

Hello Oblivion,

> We already have HTTP(S)(HttpRequest class does that) and FTP(S).

Ok.

> Have you looked at the FTP package that I wrote? (https://www.ultimatepp.org/forums/index.php?t=msg&th=889 9&goto=43053&#msg_43053)

No, but I looked now.

>- Support for extending the functionality of Ftp class, using the low-level SendCommand() method.

I think that SendCommand has other meaning then the SendCommand of the FileZilla engine.

FileZilla sends commands to the loaded executable Fzftp.exe. Fzftp.exe is derived from Putty project.

> It covers FTP, and FTPS (FTP over explicit TLS/SSL). Check the video too. It shows what it is capable of: (https://vimeo.com/214530673)

Seems good implementation. Video doesn't have sound/speech.

> FTP class I mentioned above contains an example called FtpBrowser. That example will eventually evolve into a MultiBrowser (FTP(S)/SFtp/HTTP(S)) after I finalized the SSH package. (In fact it contains a package called BrowserCtrl, which is meant to be a gui-frontend for filesystem-agnostic file browsers).

Ok. You implement what other (platforms) already have. I think it is better to implement something new, for example TOR integration. I put 'all' TOR code in 1 file (207329 lines of cpp code).

> The only protocol we seem to lack is StorJ, which is not really hard to implement. Maybe I should add it to my Todo list.

Oblivion, Is it possible to add t (for Tor) after protocol Info name, Like 'sftpt' for sftp over TOR network:

I don't know how to go to TOR implementation. Implement Libssh2 routines to the TOR implementation. I have not studie. Maybe put packets on TOR-socket?

> So while I'm not going to port filezilla to U++ (it'll need a lot of work), I'll come up with a simpler remote file browser demo.

I am trying to make a proof of concept of seperate Filezilla in three parts. GUI, engine and infrastructure code.

Greetings Jan Marco

File Attachments

1) sftp_over_tor_network.jpg, downloaded 1434 times

Subject: Re: SSH package for U++
Posted by Oblivion on Sat, 02 Dec 2017 09:44:26 GMT
View Forum Message <> Reply to Message

Hello Jan Marco,

Quote:I think that SendCommand has other meaning then the SendCommand of the FileZilla engine. FileZilla sends commands to the loaded executable Fzftp.exe. Fzftp.exe is derived from Putty project.

Sure, Ftp::SendCommand() is simply a convenient way to send control commands that aren't already covered by the FTP package.

Quote:Ok. You implement what other (platforms) already have.

Think of it this way: I am implementing what I think Ultimate++ lacks (and what I need in U++). After all, these packages are meant to be used in U++ applications.
By the way, recently I set up a GIT address where I upload 3rd party packages I wrote for U++. There are only several packages in the repo now but it will grow in time, once I clean up and publish the library I personally use: https://github.com/ismail-yilmaz/upp-components

Quote:I don't know how to go to TOR implementation. Implement Libssh2 routines to the TOR implementation. I have not studie. Maybe put packets on TOR-socket?

I have good news and (somewhat) bad news: TOR is on my todo list (with WEBDAV, OAuth2, etc...), but it isn't an easy-to-implement protocol.
However "in theory" (I didn't actually test it) it is possible to use it, without writing a whole TOR client.

There is a package called NetProxy (it is a Https/Socks4/4a/5 proxy adapter for U++) which, again, I wrote: https://www.ultimatepp.org/forums/index.php?t=msg&th=101 32&start=0&
AFAIK, Tor client (not the browser, but the background process) can be configured to be used as a SOCKS proxy.
And SSH package already supports proxied connections (through WhenProxy callback, in which you hand over the TcpSocket handle to NetProxy, and let it connect for you via a proxy server).
So if you can configure the Tor client to act as a socks 5 proxy, then, "in theory", you may be able use SSH package through it. (or any other app that is designed to support NetProxy).

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by alkema_jm on Sun, 03 Dec 2017 07:35:22 GMT
View Forum Message <> Reply to Message

Hello Oblivion,

> Think of it this way: I am implementing what I think Ultimate++ lacks (and what I need in U++). After all, these packages are meant to be used in U++ applications.

You did a very good job to make the packages for Ultimate++. I like to re use as much as possible. I don't like to 'reinvent weels'.

> By the way, recently I set up a GIT address where I upload 3rd party packages I wrote for U++.

I use github a lot to retrieve source code. All ('living') projects are on github. :p

> There are only several packages in the repo now but it will grow in time, once I clean up and publish the library I personally use: https://github.com/ismail-yilmaz/upp-components

I like Fossil program/concept (https://www.fossil-scm.org/xfer/timeline). Last two weeks of this year i will try to merge it with Ultimate++.

> I have good news and (somewhat) bad news: TOR is on my todo list (with WEBDAV, OAuth2, etc...), but it isn't an easy-to-implement protocol.

Ok. I like good news :p

> However "in theory" (I didn't actually test it) it is possible to use it, without writing a whole TOR client.

https://tor.stackexchange.com/questions/3421/route-c-through -tor-using-socks :

Tor is a socks5 proxy.
here is the socks5 rfc Protocol is https://www.ietf.org/rfc/rfc1928.txt
here is a guide to how socks5 works with tor
https://samsclass.info/122/proj/how-socks5-works.html read this, it is VERY useful

if using sockets (I assume c++ uses sockets) you will need to
1. connect to tor (127.0.0.1:9050 by default)
2. Send authentication (5,1,0) see rfc part 3
3. Receive the tor response (5,0) see rfc part 3
4. Send Client's Connection request (5,1,0,3 + host length + a binary representation of the host and port) see rfc part 4
5. receive the tor response (5,0,0,1,0,0,0,0,0,0) see rfc part 6 (there can be a bunch of errors here, so watch out)
6. Send a binary representation of a http request to tor (Tor will forward this to the destination)
7. Receive the http response (will send the header first then the web page)

> There is a package called NetProxy (it is a Https/Socks4/4a/5 proxy adapter for U++) which, again, I wrote: https://www.ultimatepp.org/forums/index.php?t=msg&th=101 32&start=0&

Ok, Looks very promising :p

So if you can configure the Tor client to act as a socks 5 proxy, then, "in theory", you may be able use SSH package through it. (or any other app that is designed to support NetProxy).

> I see in "Tor is a socks5 proxy" on
https://tor.stackexchange.com/questions/3421/route-c-through -tor-using-socks

> AFAIK, Tor client (not the browser, but the background process) can be configured to be used as a SOCKS proxy.

In de Tor logging it see "Opening Socks listener on 127.0.0.0:9050"

> And SSH package already supports proxied connections (through WhenProxy callback, in which you hand over the TcpSocket handle to NetProxy, and let it connect for you via a proxy server).

Oblivion, Must I insert NetProxy source code in Filezilla.exe or Tor.exe? N.B. My Tor.exe is the program to connect to the Tor-network. And has a sock listerer port on localhost port 9050.

Greetings Jan Marco

Appendix FileZilla client:

File Attachments
1) filezilla_screendump.gif, downloaded 1304 times

Subject: Re: SSH package for U++
Posted by alkema_jm on Mon, 04 Dec 2017 18:01:58 GMT
View Forum Message <> Reply to Message

Hello Oblivion,

>I have good news and (somewhat) bad news: TOR is on my todo list (with WEBDAV, OAuth2, etc...), but it isn't an easy-to-implement protocol.

Good news: FileZilla has a build in (general) Proxy server, which can be used by TOR

See https://forum.filezilla-project.org/viewtopic.php?t=1398 and
https://forum.filezilla-project.org/viewtopic.php?t=283

Greetings Jan Marco

File Attachments
1) FileZilla_TOR_General_proxy.jpg, downloaded 1366 times

Subject: Re: SSH package for U++
Posted by Oblivion on Mon, 04 Dec 2017 21:46:29 GMT
View Forum Message <> Reply to Message

Hello Jan Marco,

Below is a simple example demonstrating the most basic way to connect to an ssh server via TOR protocol, using SSH package. (Naturally, it requires NetProxy package):

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

using namespace Upp;

CONSOLE_APP_MAIN
{
 // This example requires a running TOR daemon.
 // Change below strings to your preferred values.

 const char* ssh_host = "dummysshhostname";
 const char* ssh_user = "dummysshusername";
 const char* ssh_pass = "dummysshpassword";
 int         ssh_port = 22;

 StdLogSetup(LOG_FILE|LOG_COUT);
 Ssh::Trace();
 NetProxy::Trace();

 SshSession session;
 session.WhenProxy = [=, &session] {
  return NetProxy(session.GetSocket(), "127.0.0.1", 9050)
            .Timeout(30000)
            .Socks5()
            .Auth("none", "none")
            .Connect(ssh_host, ssh_port);
 };

 if(session.Timeout(60000).Connect(ssh_host, ssh_port, ssh_user, ssh_pass)) {
  LOG("Successfully connected to " << ssh_host << " (over TOR)");
 }
 else
  LOG("Ssh connection via TOR failed. " << session.GetErrorDesc());
}
```

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by alkema_jm on Fri, 08 Dec 2017 10:22:40 GMT
View Forum Message <> Reply to Message

Hello Oblivion,

Thank you for the source code.
I made a flow diagram in Visio:

Greetings Jan Marco

File Attachments

1) sftp_platform_02.gif, downloaded 1300 times

Subject: Re: SSH package for U++
Posted by alkema_jm on Wed, 13 Dec 2017 16:46:17 GMT
View Forum Message <> Reply to Message

Hello Oblivion,

> So while I'm not going to port filezilla to U++ (it'll need a lot of work), I'll come up with a simpler remote file browser demo.

FileZilla engine parses filestem info depending on Operating System.

```
bool ParseAsUnix(CLine &line, CDirentry &entry, bool expect_date);
bool ParseAsDos(CLine &line, CDirentry &entry);
bool ParseAsEplf(CLine &line, CDirentry &entry);
bool ParseAsVms(CLine &line, CDirentry &entry);
bool ParseAsIbm(CLine &line, CDirentry &entry);
bool ParseOther(CLine &line, CDirentry &entry);
bool ParseAsWfFtp(CLine &line, CDirentry &entry);
bool ParseAsIBM_MVS(CLine &line, CDirentry &entry);
bool ParseAsIBM_MVS_PDS(CLine &line, CDirentry &entry);
bool ParseAsIBM_MVS_PDS2(CLine &line, CDirentry &entry);
bool ParseAsIBM_MVS_Migrated(CLine &line, CDirentry &entry);
bool ParseAsIBM_MVS_Tape(CLine &line, CDirentry &entry);
int ParseAsMlsd(CLine &line, CDirentry &entry);
bool ParseAsOS9(CLine &line, CDirentry &entry);
```

I made a Visio diagram how I see the connection between SSH en SSH-server. I want to use https://github.com/PowerShell/Win32-OpenSSH because it compilers/links in vs2015:

There are a lot of info en Queue programs. I am looking First at librdkafka, maybe there are better

ones.

https://content.pivotal.io/rabbitmq/understanding-when-to-us e-rabbitmq-or-apache-kafka

librdkafka is a C library implementation of the Apache Kafka protocol, containing both Producer and Consumer support. It was designed with message delivery reliability and high performance in mind, current figures exceed 1 million msgs/second for the producer and 3 million msgs/second for the consumer.
librdkafka is licensed under the 2-clause BSD license. https://github.com/edenhill/librdkafka

Maybe you have some advise or feedback :)

Greetings Jan Marco

File Attachments

1) SSH_server_02.gif, downloaded 1180 times

Subject: Re: SSH package for U++
Posted by Oblivion on Tue, 19 Dec 2017 13:27:07 GMT
View Forum Message <> Reply to Message

Hello Jan Marco,

Sorry I couldn't reply to you earlier.

I may be able to help but I'd need to know what you really want to achive. A server, client, or client-server?

Quote:
There are a lot of info en Queue programs. I am looking First at librdkafka, maybe there are better ones.

I'd never heeard of this one before. Could you elaborate on why you need this?

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by Oblivion on Tue, 19 Dec 2017 13:44:30 GMT
View Forum Message <> Reply to Message

Hello,

SSH package (both core classes and examples) is updated. As usual, you can find the latest package in the first message of this topic, or via below git address:

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

SSH package gained new features:

- SshSession: It is now possible to use encryption keys loaded into memory. (Load keys from mem.)

- SshSession: Host based authentication method is added.

- Ssh: TraceVerbose() method is added. This method allows full-level logging (redirection) of libsssh2 diagnostic messages.

- Documentation updated.

Examples are updated too. Two new reference example is added to the package:

- SshOverTor: Demonstrates a basic SSH connection over TOR (Requires NetProxy package and a TOR daemon)

- SshLoggingExample: Demostrates logging capabilities of SSH pakcage.

SshLoggingExample demonstrates the powerful logging mechanism of SSH package:

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

using namespace Upp;

// To activate verbose logging, set the LIBSSH2TRACE flag.
// (e.g. via TheIDE->main configuration settings)

CONSOLE_APP_MAIN
{
 StdLogSetup(LOG_COUT | LOG_FILE);

 // Ssh::Trace();

 Ssh::TraceVerbose(
```

```
//  LIBSSH2_TRACE_SOCKET    |
  LIBSSH2_TRACE_KEX       |
  LIBSSH2_TRACE_AUTH      |
  LIBSSH2_TRACE_CONN      |
//  LIBSSH2_TRACE_SCP       |
//  LIBSSH2_TRACE_SFTP      |
//  LIBSSH2_TRACE_PUBLICKEY |
  LIBSSH2_TRACE_ERROR
 );
 SshSession session;
 auto b = session.Timeout(30000).Connect("demo:password@test.rebex.net:22");
 LOG((b ? "Successfully connected to SSH2 server." : session.GetErrorDesc() << '\n'));
}
```

Reviews, patches, bug fixes, criticism, and suggestions are always appreciated.

Best regards,

Oblivion

Subject: Re: SSH package for U++
Posted by Oblivion on Sun, 07 Jan 2018 21:09:21 GMT
View Forum Message <> Reply to Message

Hello,

In the following days, SSH package will receive a huge update. There are many bug fixes and improvements (hopefully with performance gain) here and there, but the most significant changes are made to SshChannel, which is now a full-fledge, and covering SSH channel class.

In the meantime I am putting here a short video of the one of the new members of ssh channel related classes: SshShell :)

Update will contain SShShell, SShTunnel, and SshX11Tunnel classes. (Video link is down below the image)

Below video demonstrates a real-time remote shell (simulated on a local test machine).

https://vimeo.com/250031042

Best regards,
Oblivion

File Attachments
1) SshShell-Screenshot.jpg, downloaded 1182 times

Subject: Re: SSH package for U++
Posted by koldo on Mon, 08 Jan 2018 07:27:51 GMT
View Forum Message <> Reply to Message

It looks great.

I'm going to use it again soon, and I'm looking forward to it!

Subject: Re: SSH package for U++
Posted by Oblivion on Tue, 09 Jan 2018 22:15:45 GMT
View Forum Message <> Reply to Message

Hello Koldo,

I hope you'll find it useful. :)

Below screenshot (and short video) demonstrates the real-time non-blocking ssh shell with GUI integration (for the exapmle I used CodeEditor). (supports both Windows and *NIX)
"It just works". :)

Video:
https://vimeo.com/250352882

Screenshot (obviously):

Best regards,
Oblivion

File Attachments
1) ssh_shell_gui.jpg, downloaded 1168 times

Subject: Re: SSH package for U++
Posted by koldo on Wed, 10 Jan 2018 07:53:34 GMT
View Forum Message <> Reply to Message

Very didactic!

Subject: Re: SSH package for U++
Posted by Oblivion on Fri, 19 Jan 2018 13:18:32 GMT
View Forum Message <> Reply to Message

Hello everyone,

SSH package is updated. This is a major update.

2018-01-19: Alpha version 2.

SshChannel reworked. It is now more flexible, and more analogous to a tcp socket.

Scp class gained a new Put method (and its corresponding operator overload).

NEW: SShShell is added to the package. It allows GUI integration and has a "console mode" that supports both POSIX consoles and Windows command prompt.

NEW: SshTunnel class is added to the package. It allows TCP/IP and port forwarding over SSH protocol.

Various bug fixes, and improvements.

Finally the code for SSH shell (console/interactive command line interface) has landed. I managed to get it work on Windows command prompt too (ver >= XP). 8)

Examples are updated too. SshShell, SshShellNB, and SshShellGUI examples are added to the package.

SShShellGUI demonstrates a very basic SSH terminal with a non-blocking GUI.

You can always find the updated package on the first message of this topic or you can grab the code from:

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

Short videos:

SSH Shell (Demonstrating console apps, such as nano, and top):
https://vimeo.com/250031042
SSH Shell GUI (demonstrates a very basic SSH terminal with a non-blocking GUI):
https://vimeo.com/250352882

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

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by Oblivion on Sun, 21 Jan 2018 11:18:12 GMT
View Forum Message <> Reply to Message

Hello,

X11 forwarding is the last missing part of the SSH package, and it is shaping up good.

It is based on SshShell class and supports both console and GUI modes. There is even a good chance that it'll work on Windows, since the necessary foundation is alreay laid.

I'm looking into the possibility of compile-time (optional) integration with XMing or Cygwin/X.

Below short video can give you the idea. :)

https://vimeo.com/252025261

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by koldo on Sun, 21 Jan 2018 15:53:04 GMT
View Forum Message <> Reply to Message

8o Amazing!

Subject: Re: SSH package for U++

Posted by Oblivion on Sun, 28 Jan 2018 11:13:10 GMT
View Forum Message <> Reply to Message

Hello all,

X11 forwarding support (the final missing piece of SSH package) code has finally landed.

X11 support is added as an operation mode for SshShell.

I believe that we have achieved a really remarkable result here (though this is a work-in-progress), given that:

- All SSH components have a very simple, easy to use, and uniform interface that supports time-constrained blocking, non-blocking operation modes and multithreading,

- They all work on Windows (tested on 7 & 10) and POSIX-compliant operating systems, and compile on both GCC/MingGW, and MSC.

- Shell component can work simultaneously with multiple X11 forwarding (per-shell), which is AFAIK a very rare feature among the libssh2 wrappers out there.

- And all this can be achieved writing very little code! (e.g. SshX11Shell has 10 LOCs for the actual code of X11-enabled full console, and SshShellGUI has 156 LOCs which are mostly usual U++ GUI setup)

Examples directory contains SshX11Shell and re-written SshShellGUI example with X11 and multiple shell support.

Here is a screenshot:

As usual, you can find the code and examples in the first message of this topic,
or you can grab them from: https://github.com/ismail-yilmaz/upp-components/tree/master/Core/SSH

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

Best regards,
Oblivion

File Attachments

1) ShellwithX.png, downloaded 1075 times

Subject: Re: SSH package for U++
Posted by Oblivion on Mon, 02 Apr 2018 21:27:24 GMT
View Forum Message <> Reply to Message

Hello,

The new FTP package contains a simple FTP GUI example called FtpGUI. And now, here is its SFTP counterpart.
It has the same functionality. More importantly, their code are almost identical. 8)

You can compare them from the below links:

FtpGUI:
https://github.com/ismail-yilmaz/upp-components/tree/master/ Examples/FtpGUI

SFtpGUI:

https://github.com/ismail-yilmaz/upp-components/tree/master/ Examples/SFtpGUI

Screenshot:

I'm considering writing multithreaded and multiprotocol (Local, FTP/SFTP/HTTP) global file get and put functions (similar to a lightweight GIO/KIO) for U++, based on AsyncWork, and using a url scheme.
If you have any thougts about this, I'd like to hear it.

Best regards,
Oblivion

File Attachments

1) SFtpGUI.png, downloaded 1142 times

Subject: Re: SSH package for U++
Posted by Oblivion on Sat, 07 Apr 2018 14:25:49 GMT
View Forum Message <> Reply to Message

Hello all,

SSH package and examples are updated.
There are some small improvements and a small change:

- Blocking and non-blocking behaviour is now very similar to TcpSocket's.

- IsBlocking(), IsWorking() methods are added.
- WhenDo replaced with WhenWait, and WaitStep() method is added

Upcoming version will break the interface slighlty, I'm afraid. This will be a final breakage, as I am also going to declare the package a beta stage library.
Almost all of the issues are resolved, and last bits are going to be resovelved -hopefully- with the next version.
SSH package is currently very stable though.

Yet, as with the Ftp, I will also move this version of the SSH into the Attic foder, and maintain it for some time.

I am going to make the following changes:

Remove progress gate parameters in the getters and putters, and replace them with a single WhenProgress gate.
- Add WhenContent: Consumer function for incoming data transfers.

Change the Async (multithreaded) getters and putters:
- They will have progress gates with three-parameteres (id, done, total), instead of two. This proved more useful.
- They will use a URL similar to the one I use in the new version of FTP package.
- Additional variants for async functions, which will use One<Stream> and picks input data (for outgoing transfers).

- Update libssh2, as it has seen some activitiy, and gained new ciphers.

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

Best regards,
Oblivion.

Subject: Re: SSH package for U++
Posted by Oblivion on Sat, 14 Apr 2018 22:39:51 GMT
View Forum Message <> Reply to Message

Hello,

SSH package updated. It has finally reached beta version.

2018-04-15: Consumer function support added to SFtp and SshChannel classes.
GetWaitStep() method is added to Ssh class.
Multithreaded methods rewritten.

It is now possible to use consumer functions. A reference example demonstrating this behavious is added, accordingly.
Also, multithreaded functions are rewritten. They now use a three-parameter progress gate.

Consumer function example (SFtpConsumerGet):

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

using namespace Upp;

CONSOLE_APP_MAIN
{
 StdLogSetup(LOG_COUT|LOG_FILE);
// Ssh::Trace();

 const char *file = "/pub/example/readme.txt";

 String data;

 SshSession session;
 if(session.Timeout(30000).Connect("demo:password@test.rebex.net:22")) {
  auto sftp = session.CreateSFtp();
  sftp.WhenContent = [&data](const void *buf, int len)
  {
   data.Cat(static_cast<const char*>(buf), len);
  };

  sftp.Get(file);
  LOG((!sftp.IsError() ? data : sftp.GetErrorDesc()));
 }
 else
  LOG(session.GetErrorDesc());
}
```

Below you can find the latest package and the GIT address where you can always get the latest version.

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

Examples: https://github.com/ismail-yilmaz/upp-components/tree/master/ Examples
Older Version: https://github.com/ismail-yilmaz/upp-components/tree/master/ Attic/SSH

Please feel free to comment on it. Bug reports, reviews, criticism, etc. are appreciated.

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by Oblivion on Sat, 21 Apr 2018 06:12:22 GMT
View Forum Message <> Reply to Message

Hello,

SSH package and its reference examples are updated.

Two new multithreaded transfer methods are added to the package:

static AsyncWork<void> SFtp::AsyncConsumerGet(SshSession& session,
const String& path, Event<int64, const void*, int> consumer)
static AsyncWork<void> Scp::AsyncConsumerGet(SshSession& session,
const String& path, Event<int64, const void*, int> consumer)

These asynchronous methods use a consumer function to download data.

Also, two new reference examples are added to the package:

- SFtpConsumerGetMT: Demonstrates the usage of a consumer function for SFtp download, using a worker thread.
- SshPolymorphismNB: Demonstrates the polymorphism capability of SSH channels in non-blocking mode.

You can always get the latest version from the below git adresses.

GIT repo: https://github.com/ismail-yilmaz/upp-components/tree/master/ Core/SSH
Examples: https://github.com/ismail-yilmaz/upp-components/tree/master/ Examples
Older Version: https://github.com/ismail-yilmaz/upp-components/tree/master/ Attic/SSH

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by Oblivion on Fri, 04 May 2018 20:08:33 GMT
View Forum Message <> Reply to Message

Hello

SSH package is updated.

SSH:

2018-05-04: Ssh::GetWaitEvents() fixed.
SshTunnel::Validate() fixed.

Reference examples added to the package:

SshTunnelExample: Demonstrates the basic SSH tunneling (as tunnel/server) in blocking mode.

Note that the SSH tunnel example requires upp/reference/SocketClient and upp/reference/SocketServer examples.
it acts as a SSH server between the socket client and server. Although the example demonstrates one of the basic tunneling capabilities of the SshTunnel class,
very complex SSH tunnels can be built using it.

Code:

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

using namespace Upp;

// This example requires upp/reference/SocketServer and upp/reference/SocketClient examples.
// SocketClient: Set the port number to 3215.
//
// |SocketClient (client)|<---> |SshTunnelExample (tunnel/server)| <---> |SocketClient (server)|

bool SocketSendRecv(String& packet)
{
 TcpSocket s;
 if(!s.Connect("127.0.0.1", 3214)) {
  LOG("SocketSend(): " << s.GetErrorDesc());
  return false;
 }
 if(!s.PutAll(packet + '\n'))
  return false;
 packet = s.GetLine();
```

```
 return !packet.IsEmpty();
}
void StartTunnel(SshSession& session)
{
 SshTunnel listener(session);
 if(!listener.Listen(3215, 5)) {
  LOG("StartTunnel(): " << listener.GetErrorDesc());
  return;
 }
 LOG("SSH tunnel (server mode): Waiting for the requests to be tunneled...");
 for(;;) {
  SshTunnel tunnel(session);
  if(!tunnel.Accept(listener)) {
   LOG("StartTunnel(): " << tunnel.GetErrorDesc());
   return;
  }
  auto data = tunnel.GetLine();
  LOG("Tunneled Request: " << data);
  if(!data.IsEmpty() && SocketSendRecv(data)) {
   LOG("Tunneled Response: " << data);
   tunnel.Put(data + '\n');
  }
 }
}

CONSOLE_APP_MAIN
{
 StdLogSetup(LOG_FILE | LOG_COUT);
// Ssh::Trace();

 SshSession session;
 if(session.Timeout(30000).Connect("username:password@localhost:22")) {
  StartTunnel(session.Timeout(Null));
  return;
 }
 LOG(session.GetErrorDesc());
}
```

Below is the GIT address where you can always get the latest version.

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

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by Oblivion on Sat, 16 Jun 2018 17:09:21 GMT
View Forum Message <> Reply to Message

Hello,

SSH package and its examples are updated.
I highly recommentd updating the package, since it contains a critical fix.

2018-06-15: Critical fix: SshChannel and SFtp data read and write methods fixed: The recently introduced
socket wait mechanism was causing a constant I/O blocking. This is now fixed.
Cosmetics & cleanup.

2018-06-06: SshShell: Console loop fixed.

You can find the updated packages in the first message of this topic, or at the below git address:

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

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

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by mirek on Tue, 10 Jul 2018 12:37:28 GMT
View Forum Message <> Reply to Message

https://www.ultimatepp.org/forums/index.php?t=msg&goto=5 0064&#msg_50064

libeay32MTd ssleay32MTd are now unavailable for U++ setup....

(you have these configured in SSH).

Subject: Re: SSH package for U++
Posted by mirek on Tue, 10 Jul 2018 13:38:17 GMT
View Forum Message <> Reply to Message

Also, it looks like with new SSL, you need to add crypt32.lib for win32...

(Sending ssh.upp that seems to work now).

File Attachments

1) SSH.upp, downloaded 340 times

Subject: Re: SSH package for U++
Posted by mirek on Tue, 10 Jul 2018 13:42:34 GMT
View Forum Message <> Reply to Message

SSH looks pretty reasonable. Can I move to Core/SSH? :)

That said, I find myself sort of cofusing the order of arguments sometimes:

Put(Stream& in, const String& path);

I would expect path to be the first argument here... Do you insist on this order? :)

Mirek

Subject: Re: SSH package for U++
Posted by Oblivion on Wed, 11 Jul 2018 06:35:12 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Quote:
SSH looks pretty reasonable. Can I move to Core/SSH?

Of course! :)

Quote:Put(Stream& in, const String& path);

I would expect path to be the first argument here... Do you insist on this order?

That's because I use "[source], [destination]..." order. I'd prefer it stay that way but I don't insist on it. It can be changed.

What's left (none of them are critical, and can be added later via patches):

1) Implementation document (giving an overview of the package and its implementation details).
2) MT server (listener) mode for SshTunnel.
3) Refactoring of Ssh (core) class.

I am going to test the package with the latest build of the U++ on Windows, tonight.

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by mirek on Tue, 31 Jul 2018 09:07:49 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Wed, 11 July 2018 08:35Hello Mirek,

Quote:
SSH looks pretty reasonable. Can I move to Core/SSH?

Of course! :)

Quote:Put(Stream& in, const String& path);

I would expect path to be the first argument here... Do you insist on this order?

That's because I use "[source], [destination]..." order. I'd prefer it stay that way but I don't insist on it. It can be changed.

Whether the "source, target" or "target, source" is more natural is certainly debatable, however please notice that since the dawn of computer languages the custom is to write target = source (also see strcpy etc... :) That said, where handle or path is involved, I would probably like to have it as first parameter always.

Besides, it does not seem to be consistent anyway:

```
bool            Get(SFtpHandle* handle, Stream& out);
bool            Put(SFtpHandle* handle, Stream& in);
```

Now I am only scrathich the surface, mostly being interested in SFTP for now. Anyway, seeing that interface, I would probably like to change following things:

SFtpHandle* -> SFtpHandle - if something is "HANDLE", it should not be a pointer to handle.

Instead of:

auto sftp = session.CreateSFtp();

I would like to have

SFtp sftp(session);

Besides, I know that there is some movement to use 'auto' everywhere, but I strongly disagree with that, ESPECIALLY in reference examples.

More important: I thing I would remove whole bunch of Gets and Puts, leave only handle variant in sftp and then add SFtpStream instead. With methods provided in SFtp, it should be easy to do, and it would mostly remove "source/destination controversy".

Interestingly, it looks like the most important Gets / Puts are missing there

int Get(SFtpHandle h, void *ptr, int size);
bool Put(SFtpHandle h, const void *ptr, int size);

Anyway, it is now moved to Core/SSH and single example is now in reference - that is another issue I have not solved for now, there is a lot of examples and they really make sense, so maybe I will put them gradually to SSH subfolder.

I am giving you write access to Core/SSH and reference.

Mirek

Subject: Re: SSH package for U++
Posted by Oblivion on Fri, 03 Aug 2018 15:34:55 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Quote:Instead of:

auto sftp = session.CreateSFtp();

I would like to have

SFtp sftp(session);

This is already possible.

It's just that I preferred the method versions over the constructor in the examples.

Quote:
More important: I thing I would remove whole bunch of Gets and Puts.

Interestingly, it looks like the most important Gets / Puts are missing there

int Get(SFtpHandle h, void *ptr, int size);
bool Put(SFtpHandle h, const void *ptr, int size);

Ok, since the next version of the SSH package might end up to be the official ssh plugin for U++, I can change the data I/O api at this point.
I'll re-write the gets and puts. They'll be more aligned with the TcpSocket's.

Quote:
... then add SFtpStream instead. With methods provided in SFtp, it should be easy to do, and it would mostly remove "source/destination controversy".

This is a good and interesting idea. :)
I'll see what I can do.

Ok then, I was going to update the package, but I'll delay the next update, and first come up with the changes you've asked.

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by mirek on Fri, 03 Aug 2018 16:03:09 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Fri, 03 August 2018 17:34Hello Mirek,
Ok then, I was going to update the package, but I'll delay the next update, and first come up with the changes you've asked.

Note that it is now part of U++, so please it is preferable that any changes you do are done in U++ svn. You should have write rights there.

(OTOH, if you just want to continue in GIT, I will manage.. :)

Mirek

Subject: Re: SSH package for U++
Posted by Oblivion on Fri, 03 Aug 2018 16:44:25 GMT
View Forum Message <> Reply to Message

Quote:
Note that it is now part of U++, so please it is preferable that any changes you do are done in U++ svn. You should have write rights there.

Ah, OK.
I'm on a vacation now. I didn't have the time to check the svn repository. I'll start committing the changes within next week.

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by Oblivion on Tue, 07 Aug 2018 21:32:41 GMT
View Forum Message <> Reply to Message

Hello Mirek,

I'll start committing the changes in a couple of days. But before I start I'd like to clear some points and ask for your opinion on some points.

Quote:
SFtpHandle* -> SFtpHandle - if something is "HANDLE", it should not be a pointer to handle.

The thing is, libbsh2 itself returns a pointer to a handle (SFtpHandle is actually an alias for LIBSSH2_SFTP_HANDLE)
I'm reluctant to change that pointer into a real handle as it'll make the things slightly more complex.
However, if we must change the naming I would suggest we use "SFtpObject" instead, since we are dealing with remote file system objects here.

Quote:
Interestingly, it looks like the most important Gets / Puts are missing there

```
int Get(SFtpHandle h, void *ptr, int size);
bool Put(SFtpHandle h, const void *ptr, int size);
```

Easy to implement. I'll propose:

```
int       Get(SFtpObject* obj, void* buffer, int size);
String     Get(const String& path, int size, int64 offset = 0);
String     GetAll(const String& path);

int       Put(SFtpObject* obj, const void* buffer, int size);
int       Put(const String& path, const String& data, int size, dword flags, long mode);
int       Put(const String& path, const String& data, int size, int64 offset = 0);
bool       PutAll(const String& path, const String& data);
```

As for the stream versions of Gets and Puts:
Stream versions come very handy when transferring files > 2GB, and processing the I/O immediately (e.g when filtering).

I've thought about adding SFtpStream variants(eg. SFtpFileIn, SFtpFileOut, etc.) to the package, as you suggessted.
But in the end I concluded that they'll only add another layer of abstraction which can be confusing (as you know, SSH package already has a plenty of classes). Thus I propose to simply rename them and change their parameter order.

E.g.

```
bool GetStream(const String& path, Stream& out);
bool PutStream(const String& path, Stream& in);
// And other variants...
```

About the reference examples:

My idea is to create a reference -console- example for each class (SFtp, Scp, SshExec, SshShell, SshTunnel), and have them each contain separate and basic blocking, non-blocking, and multitihreaded code as functions that can be selected/compiled using preprocessor directives.(defs,e.g. BLOCKING, NONBLOCKING, MULTITHREADED). And use these examples as tutorial in Topic++.

What do you think?

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by mirek on Wed, 08 Aug 2018 08:56:47 GMT
View Forum Message <> Reply to Message

Well, I have already did SFtpHandle* -> SFtpHandle with no ill effects.

Now I am trying to deal with "Cmd" and async operations.

I undestand the appeal, but I have to say that the result is sort of confusing. E.g. I have implemented

int Get(SFtpObject* obj, void* buffer, int size);

but it is clearly incompatible with this sort of async operations. All those NULL handles, implicit results etc make me uneasy.

Meanwhile, I have started to checking coming coroutines support (probably C++20) - on the first look, these look like "queue done right".

So I guess for now, I will try to pretend that those complex Cmd / ComplexCmd "nonblocking" operations are not there, probably putting assert to make that sure for methods like "Get" and hope to do all that right with co_wait / co_return in the future.

In related news, I have also fixed GetWaitEvents

```
dword Ssh::GetWaitEvents()
{
 ssh->events = 0;
 if(ssh->socket && ssh->session)
  ssh->events = libssh2_session_block_directions(ssh->session);
 return !!(ssh->events & LIBSSH2_SESSION_BLOCK_INBOUND) * WAIT_READ +
        !!(ssh->events & LIBSSH2_SESSION_BLOCK_OUTBOUND) * WAIT_WRITE;
}
```

I am also thinking that perhaps SFtp should be (derived from) SshSession. I think it is unlikely that sharing SshSession for several protocols is all that important.

I undestand the appeal, but I have to say that the result is sort of confusing. E.g. I have implemented

int Get(SFtpObject* obj, void* buffer, int size);

but it is clearly incompatible with this sort of async operations.

Well, I don't plan that method to be async (MT). IMO Async methods (that have SFtp:Asyncxxx prefix) should either use streams or consumer function (Event<const void*, int>), which will have the same effect anyway.

Asynchronous (MT) operations on ssh is somewhat complicated. I had to make some compromises for the sake of simplicity and maintainabilily, but first I have to see tha changes you've made to the code, then I'll write a summary of its design and explain those "odd" choices (it's a complex beast but works fine.)

So I guess for now, I will try to pretend that those complex Cmd / ComplexCmd "nonblocking" operations are not there, p

Those methods are the parts that I am not happy with, either :)
I have a plan and a test code to "fix" that ugliness, but actual refactoring will come later.

All those NULL handles, implicit results etc make me uneasy.

Most of them are, I believe, taken care of and contained. Implicit results are there, but shouldn't pose any real danger (Not that Ic can see of, at least). They can, and hopefully will, be reduced (another TODO for me,connected with Cmd/ComplexCmd pair).

In related news, I have also fixed GetWaitEvents

Did it make any difference? Because IIRC values of those upp enums and the defines of libssh2 are the same.

Quote:
I am also thinking that perhaps SFtp should be (derived from) SshSession. I think it is unlikely that sharing SshSession for several protocols is all that important.

I repectfully disagree. Servers usually limit the number of sessions. (e.g. the servers I work with allows only five sessions from the same user.) OTOH, there is no channel limit. (they are only limited with bandwidth). Also this way we will give the developers flexibility. For example,If I have the ability to use Scp to transfer files while browsing and manipulating file system objects with SFtp, I'll prefer Scp for transfers, since it is relatively faster.

Also I have a plan to override libssh2 raw data read and write methods (it allows it) and use TcpSocket directly, and move Wait() there.

Best regards,
Oblivion.

Subject: Re: SSH package for U++
Posted by mirek on Wed, 08 Aug 2018 11:28:51 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Wed, 08 August 2018 11:22
Those methods are the parts that I am not happy with, either :)
I have a plan and a test code to "fix" that ugliness, but actual refactoring will come later.

Is there any real-world example where you would need this kind of nonblocking behaviour?

All I can came up is some code that communicates with thousands of ssh servers at once. Looks very unlikely to me....

Quote:
Did it make any difference? Because IIRC values of those upp enums and the defines of libssh2 are the same.

Are they documented to be the same?

Mirek

Subject: Re: SSH package for U++
Posted by Oblivion on Wed, 08 Aug 2018 12:16:38 GMT
View Forum Message <> Reply to Message

Is there any real-world example where you would need this kind of nonblocking behaviour?

All I can came up is some code that communicates with thousands of ssh servers at once. Looks very unlikely to me....

I use it in an app and for a limited number of ssh channels (usually 10-20).
But frankly, that code remains before the the CoWork improvements and the arrival of AsyncWork.
Nowadays in most such cases I use the async methods.

A short technical info:

- Ssh-based classes use a single callable target queue (FIFO), relying on deferred execution.

- Call to the queue elements are protected by a single static mutex (using INTERLOCKED).

- This allows easy maintenance but has a negative impact on the asynchronicity (it is not really possible to achieve %100 asynchronous operations with the design of libssh2 anyway. Yet the performance gains can be up to %30 in MT mode)

- Queue is populated by two interrelated methods: Cmd/ComplexCmd.
  Cmd: Adds a single ananymous function to the callable target queue. In blocking mode it executes the function immediately. In non-blocking mode it defers the execution.
  ComplexCmd: Firs and foremos,t this method acts as a nest for other Cmds. and other nests. This way it is possible to execute command chains in a consistent way (as if they are a single CMD).

Certainly you are not comfortable with the current implementation. (Besides, I haven't tested it yet bu as far as I can see your Get() implementation in svn won't work in non-blocking mode)

I have a new proposal: What if I get rid of queue mechanism and rewrite the package with only blocking mode and optional async transfer methods(using AsyncWork, and naming them agein SFtp::Asyncxxx)?
It won't take more than a week for me to come up with a working SSH package and the existing public API wont change much (only the NB helpers will be gone).
Besides its SC will be lot cleaner.

In fact, I had a working prototype of this (was using my Job class) I ditched in favour of the existing version.

As to your Get implementation:

```
int SFtp::Get(SFtpHandle handle, void *ptr_, int size0)
{
 int done = 0;                          // <- Can't be used in non-blocking mode.
 char *ptr = (char *)ptr_;
 Cmd(SFTP_START, [=, &done]() mutable {
  int size = size0;
  if(OpCode() == SFTP_START) {
   if(FStat(HANDLE(handle), *sftp->finfo, false) <= 0) OpCode() = SFTP_GET; // <- This is for
higher-level api. should be removed.
   else return false;
  }
  while(size) {
   int rc = libssh2_sftp_read(HANDLE(handle), ptr, min(size, ssh->chunk_size));
   if(rc < 0) {
    if(!WouldBlock(rc))
     SetError(rc);
    return false;
   }
   else {
    if(rc == 0)
     break;
    size -= rc;
    done += rc;
    if(WhenProgress(done, size0))
     SetError(-1, "Read aborted.");
    ssh->start_time = msecs();
   }
  }
  LLOG(Format("%d of %d bytes successfully read.", done, size0));
  return true;
 });
 return done;
}
```

I haven't tested this yet, but it shouldn't work in non-blocking mode. (because the execution will be deferred (Get will immediately return) and there is a local variable ("done"))

Quote:
Are they documented to be the same?

Nope, its my fault.
Best regards,
Oblivion.

Subject: Re: SSH package for U++
Posted by mirek on Thu, 09 Aug 2018 09:54:01 GMT
View Forum Message <> Reply to Message

Quote:
I use it in an app and for a limited number of ssh channels (usually 10-20).
But frankly, that code remains before the the CoWork improvements and the arrival of AsyncWork.
Nowadays in most such cases I use the async methods.

OK, no need to have fully async mode for this....

Quote:
I have a new proposal: What if I get rid of queue mechanism and rewrite the package with only blocking mode and optional async transfer methods(using AsyncWork, and naming them agein SFtp::Asyncxxx)?
It won't take more than a week for me to come up with a working SSH package and the existing public API wont change much (only the NB helpers will be gone).
Besides its SC will be lot cleaner.

Definitely.

Quote:
As to your Get implementation:

I haven't tested this yet, but it shouldn't work in non-blocking mode. (because the execution will be deferred (Get will immediately return) and there is a local variable ("done"))

Sure, as I said that was the point where I decided that fully non-blocking mode is "blocking" this kind of interface.

BTW, digging deeper into the code

```
bool SshChannel::Lock()
{
 if(*lock == 0) {
  LLOG("Channel serialization lock acquired.");
  *lock = ssh->oid;
 }
 return *lock == ssh->oid;
}
```

I think there is a race condition here - two threads can obtain this lock simultaneously. Now I am not sure whether is this supposed to be MT safe, but if not, why atomic, right?

Mirek

Subject: Re: SSH package for U++
Posted by mirek on Thu, 09 Aug 2018 10:00:38 GMT
View Forum Message <> Reply to Message

I am also pretty ambivalent about all those static AsyncWork methods. I think these are better left to client code. Especially if we abandon the non-blocking mode.

Mirek

Subject: Re: SSH package for U++
Posted by Oblivion on Thu, 09 Aug 2018 14:24:47 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Quote:
Sure, as I said that was the point where I decided that fully non-blocking mode is "blocking" this kind of interface.

I'm sorry but I really don't understand this one.
Here is the snippet of the working version of the same method (bot in blocking and non-blocking mode) from the now-cancelled-update:

```
int SFtp::Read(SFtpHandle handle, Event<const void*, int>&& consumer, int size) // Data read engine.
{
	int sz = min(size, ssh->chunksize)
  Buffer<char> buffer(sz);
 int rc = libssh2_sftp_read(HANDLE(handle), buffer, sz);
 if(!WouldBlock(rc) && rc < 0)
  SetError(rc);
 if(rc > 0) {
  consumer(buffer, rc);
  sftp->done += rc;
		if(WhenProgress(sftp->done, size))
```

```
   SetError(-1, "Read aborted.");
  ssh->start_time = msecs();
 }
 return rc;
}

int SFtp::Get(SFtpHandle* handle, void* buffer, int size)
{

        Clear();
  Cmd(SFTP_GET, [=]() mutable {
  int rc = Read(
   handle,
   [=](const void* p, int sz)
   {
     if(!buffer)
      SetError(-1, "Invalid pointer to read buffer");
    memcpy((char*)(buffer + sftp->done), (char*)p, sz);
   },
   size
  );
  if(rc >= 0)
   sftp->value = sftp->done;
  return rc == 0 || sftp->done == size;
 });
 return sftp->done;
}
```

This works as expected. Note that for any kind of get method all you have to do is simply wrap the "engine" to suit your needs.

Quote:
I think there is a race condition here - two threads can obtain this lock simultaneously. Now I am not sure whether is this supposed to be MT safe, but if not, why atomic, right?

Yep. You see, an ssh shell is a complex environment. You cannot initalize multiple shells, and/or exec channels at once (I don't want to go into details here). This is a limitation of the libssh2. Their initialization have to be in some way serialized. These so-called Lock/Unlock methods handle that serialization when multiple shells or execs were started while maintaining a single threaded non-blocking and/or multithreaded asynchronous initalization. This is what makes multiple shells or exec channels at once (and the SshShellGUI example, for that matter) possible. I added the lock for NB and started to convert it into a thread-safe version. But the idea was yet to be finalized. In the end (in the now-cancelled-update) I decided to go with a RWMutex instead.

Quote:
I am also pretty ambivalent about all those static AsyncWork methods. I think these are better left to client code. Especially if we abandon the non-blocking mode.

Ok, this is not something I would object. Once the new SSH package is done I'll write a small complementary package with a handful of MT convenience functions only and maintain it in my git repo.

To sum up:

If you find it reasonable, I'll rewrite the SSH package with only blocking mode in mind (but with neceasary thread safety, and add its components gradually).

But as I already wrote in my previous messages, let us not change the "single session-multiple channels model" and not exclude any components.
When you start working in environments that rely on ssh ecosystem, as I do, you eventually end up working with ssh tunnesl, exec, and shells.
Besides the current shell implementation we have is AFAIK one of its kind. :) I don't know any other open source shell component that does what Upp::SshShell does.
Hence it has educational and advertorial value. I mean, If you look up on stackoverflow or other relevant sites you'll see that even a barebone practical ssh shell implementation with libssh2, and also libssh is a mystery to people, let alone a shell that can have running multiple instances at the same time and that even works under Windows dumb-console. We can use this to promote U++, by writing a tutorial, demonstrating the shell in, say, codeproject.

Is this OK?

If so, I'll start writing the new code and commit the changed package within next week.

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by mirek on Thu, 09 Aug 2018 14:49:55 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Thu, 09 August 2018 16:24Hello Mirek,

Quote:
Sure, as I said that was the point where I decided that fully non-blocking mode is "blocking" this kind of interface.

I'm sorry but I really don't understand this one.

Well, if you are about to have such Get in the interface, you expect it to be usable with multiple streams at the same time. Storing into single ftp->done is no go then.

Really, let us wait for co_await and do it right then :)

Quote:
If you find it reasonable, I'll rewrite the SSH package with only blocking mode in mind (but with neceasary thread safety, and add its components gradually).

"Pseudoblocking" - we still would like to have WhenWait and Abort (afaik it is now named "Cancel", but TcpSocket is using Abort, so maybe it should have the same name). WhenWait and Abort will allow GUI around it.... (That said, I think each channel should have its own WhenWait).

In reality, I do not think that there is a lot of new things to develop. This is mostly simplifying and removing.... Probaly Cmd will get simplified, ComplexCmd probably can be removed.

I have today developed SFtpStream (based on new blocking Get/Put) and tried to add a "new schema" SaveFile / LoadFile. It is a good feeling to load file over sftp line by line :) Anyway, but supporting the standard stream, most of those specialised methods become unncessarry IMO.

Quote:
But as I already wrote in my previous messages, let us not change the "single session-multiple channels model" and not exclude any components.

Yes, after you have explained the reason, I agree. (Well, maybe it could have "private Ssh session" and both modes, but probably not worth it)

Quote:
Is this OK?

Absolutely, except I might want to work on it a bit too :) So please at least check what happens in svn.

Mirek

Subject: Re: SSH package for U++
Posted by Oblivion on Thu, 09 Aug 2018 15:03:20 GMT
View Forum Message <> Reply to Message

"Pseudoblocking" - we still would like to have WhenWait and Abort (afaik it is now named "Cancel", but TcpSocket is using Abort, so maybe it should have the same name). WhenWait and Abort will allow GUI around it.... (That said, I think each channel should have its own WhenWait).

Of course, otherwise we cant even have optional MT or even multiple channels. :) It will be blocking from the user-POV (similar to TcpSocket).

Well, if you are about to have such Get in the interface, you expect it to be usable with multiple streams at the same time. Storing into single ftp->done is no go then.
Really, let us wait for co_await and do it right then

Sure, let's wait. :) But I really don't understand the problem. Can you elaborate a little mode if you don't mind?
In an ssh system you don't use a single sftp channel for multiple/concurrent transfers. Channels are disposable. This is even encouraged. So each byte counter (done) is used for a single transfer. Then it gets cleared. We do multiple Sftp transfers with multiple sftp sessions.

Quote:
Absolutely, except I might want to work on it a bit too Smile So please at least check what happens in svn.

Of course, It's taking some time on my side because I'm thinking on the new design :)

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by Oblivion on Thu, 09 Aug 2018 15:59:22 GMT
View Forum Message <> Reply to Message

WhenWait and Abort will allow GUI around it.... (That said, I think each channel should have its own WhenWait).

In the previous version, we had WhenWait for each object. But in the end I found it unnecessary since we have a single socket.
Besides, I was planning to use TcpSocket for the next iteration of SSH package. It is possible to

use TcpSocket directly with libssh2 (libssh2 have definable callbacks for raw socket read/write functions. I was planning direcly use TcpSocket's Get and Put and move "Wait()" into those callbacks. In the current version each Ssh object has its own Wait method, which is somewhat inefficient.)

Quote:
In reality, I do not think that there is a lot of new things to develop. This is mostly simplifying and removing.... Probaly Cmd will get simplified, ComplexCmd probably can be removed.

I agree. In the blocking prototyle I'd ditched in favour of the current version I was using Cmd again, but only to execute the command immediately (no queueing).
It was the same except no queeue. This way we can also preserve the thread safety and error reporting through exceptions (in a single method) we'll need.
I'm sure that design will allow us to have a easliy restructurable base when the new co_await arrives. (Proof: this version is an improvisation on that design.)

Quote:
I have today developed SFtpStream

Great! Since we have no non-blocking mode, I have no objections left to this addition. It'll work as expected.

I'll better get started. :)

Best regards,
Oblivion.

Subject: Re: SSH package for U++
Posted by mirek on Thu, 09 Aug 2018 16:35:49 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Thu, 09 August 2018 16:24
Yep. You see, an ssh shell is a complex environment. You cannot initalize multiple shells, and/or exec channels at once (I don't want to go into details here). This is a limitation of the libssh2.

I would like to learn a bit about this. Can you give me some link(s)?

Mirek

Subject: Re: SSH package for U++
Posted by Oblivion on Sun, 12 Aug 2018 10:51:11 GMT

Hello Mirek,

I've commited the first party of updates.

A summary:

- Ssh, SshSession, and SFtp classes are refactored.

- Non-blocking mode and multithreaded methods are removed.

- The package now uses a "pseudo-blocking" technique, similar to TcpSocket's.

- Accordingly, the Ssh::Cmd and Ssh::ComplexCmd methods are removed in favor of Ssh::Run and Ssh::Do() commands.
  Run() command is the new command execution encgine. It'll hopefully take care of the possible thread safety problems.
  Do() method is protected by a single static mutex, and should be thread-safe. Multithreaded execution should be possible.

- Abort mechanism refactored.

- Sftp::Read and SFtp::Write methods are further simplified.

- SFtp::WhenProgress method is removed in favour of GetDone() method (as the esisting WhenProgress method is now pretty useless).

- SshChannel and its derivatives are currently disabled. They will be re-added gradually.

I believe the new code can be easily maintained and changed in the future. (if we plan to take advantage of coroutines, etc.)

As for the SFtpStream and its variants: They are quite handy indeed. Thanks!

Quote:
I would like to learn a bit about this. Can you give me some link(s)?

Sorry, I should've phrase my words better: SSH and its relevant protocols are just fine. The problem I mentioned arises with implementations (on server or client side).
As you know, ssh is inherently an asynchronous communication protocol relying on channel multiplexing. So both servers and clients need to keep track of every package they sent and received.
In some cases where multiple channels are requested at the same time (be it in non-blocking-ST

or in MT) channel requests fail with "Failed waiting for channel success" message.
The reason seems to be that SSH_CHANNEL_SUCCESS message for some channels are either somehow get lost in the process or not delivered in due time.
(Yet we get a valid handle for the requested channel from libssh2. So while we got a valid channel handle in the end, the subsequent calls would still fail.)
This is possibly a result of a race condition (at implementaion level).

I wrote that locking/unlocking mechanism as a workaround for this issue. Basically it functioned as a crude bu effective "WouldBlock()".
It allowed us to maintain the non-blocking behaviour while sequentially initalizing the requested channels.

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by Oblivion on Thu, 30 Aug 2018 05:27:32 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Before I commit the latest update to the code, I'd like to know if adding a "SFtpFileSystemInfo()" (non static) would be ok? It'll be an example of remote file system representation. I think FileSel, or any other file browser can benefit this. Also I can write topic docs for FileSystemInfo() as I find it a useful tool (unless it is depreceated, of course).

Best regards,
Oblivion

Subject: Re: SSH package for U++
Posted by mirek on Thu, 30 Aug 2018 07:04:16 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Thu, 30 August 2018 07:27Hello Mirek,

Before I commit the latest update to the code, I'd like to know if adding a "SFtpFileSystemInfo()" (non static) would be ok? It'll be an example of remote file system representation. I think FileSel, or any other file browser can benefit this. Also I can write topic docs for FileSystemInfo() as I find it a useful tool (unless it is depreceated, of course).

Best regards,
Oblivion

At this point I see no problem.

Subject: Re: SSH package for U++
Posted by Oblivion on Thu, 30 Aug 2018 22:12:02 GMT
View Forum Message <> Reply to Message

Hello Mirek,

I've committed the changes. Package now includes a SFtpFileSystemInfo class as an experimental feature. It has its rough edges, and is to be refined, but it works. :)

Below code is a SFtpGet to test the experimental class:

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

using namespace Upp;

GUI_APP_MAIN
{
 StdLogSetup(LOG_FILE);
// Ssh::Trace();

 SshSession session;
 if(session.Timeout(30000).Connect("demo:password@test.rebex.net:22")) {
  SFtp sftp(session);
  SFtpFileSystemInfo fsi(sftp);
  FileSel fs;
  fs.Filesystem(fsi);
  if(fs.BaseDir("/").ExecuteOpen()) {
   String file = fs.Get();
   Progress pi(nullptr, file);
   sftp.WhenProgress = [=, &pi] (int64 done, int64 total){
    pi.SetText(Format(t_("%1:s of %2:s is transferred"),
    FormatFileSize(done),
    FormatFileSize(total)));
    return pi.SetCanceled(int(done), int(total));
   };
   pi.Title(t_("Downloading ") << GetFileName(file));
   pi.Create();
   String s = sftp.LoadFile(file);
   if(sftp.IsError())
    ErrorOK(DeQtf(sftp.GetErrorDesc()));
  }
 }
 else
  ErrorOK(DeQtf(session.GetErrorDesc()));
}
```

Screenshot:

It's not a 100% integration (at least, not yet), but it can be very useful.

Best regards,
Oblivion

File Attachments

1) SFtpGetFileSel.png, downloaded 707 times

Subject: Re: SSH package for U++
Posted by Oblivion on Sun, 02 Sep 2018 19:59:04 GMT
View Forum Message <> Reply to Message

Hello Mirek,

The first official version of SSH package is almost done. Currently I'm stress-testing it (mostly for possible multithreading issues).
In the meantime, I rewrote the SFtpGui example and renamed it as SFtpBrowser.

I was going to commit it via SVN for review but it is somewhat lengthy for a "reference" example. (420 LOCs in cpp, mostly gui building)
It think it would be better included in uppsrc/examples. And as I don't have SVN access to examples directory, I'm uploading it here.

This example demonstrates:

- Both password and public key authenticationa.
- SSH gui integration.
- Using a slave sftp channel for file D/Us.
- Browsing an SFtp server.
- Basic SFtp commands (download, uploadr, rename, delete, mkdir).

I didn't include multithreading. I also wrote a multithreaded version of the browser with a download queue. Although it is a cool example, I'm not sure if it'll fit nicely in reference or examples section. So it'll be available at my git repo, along with the new version of GUI integrated ssh shell example.

Also, I'll improve the SFtpFileSystemInfo. It works well, but while I was studying the FileSel code, I've noticed that FileList/Load() function (FileSel.cpp, ln: 467-540) disregards the last modification time of files.
Is this intentional?
I think it should include that information too.

Best regards,

Oblivion

File Attachments

1) SFtpBrowser.zip, downloaded 386 times

Subject: Re: SSH package for U++
Posted by Oblivion on Wed, 31 Oct 2018 10:00:52 GMT
View Forum Message <> Reply to Message

Hello everyone,

SSH package is updated. SFtpFileSystemInfo class is now officially a part of SSH package.

What is SFtpFileSystemInfo, anyway?

SFtpFileSystemInfo helper class is a FileSystemInfo adapter for the SFtp objects.
It allows SFtp objects to be used with FileSystemInfo class that provides uniform access to folder hierarchies in a file-system agnostic ("transparent") way.

SshBasics reference examples package also reflects this addition. You can check it from the UPP nightly builds later today, when it is synced.

I'd appreciate bug reports, feedback, etc.

Best regards,
Oblivion.

Subject: Re: SSH package for U++
Posted by Oblivion on Sun, 04 Nov 2018 18:55:14 GMT
View Forum Message <> Reply to Message

Hello,

A simple GUI example is added to the SSH reference ezamples: SFtpFileSel.

This example demonstrates FileSel integration of SFtp class, using the new FileSystemInfo interface.

It can be found in the uppsrc/reference examples directory, after the today's nightly build are synced.

Screenshot:

Best regards,
Oblivion

File Attachments

1) SFtpFileSel.png, downloaded 568 times

Subject: Re: SSH package for U++
Posted by Oblivion on Fri, 22 Mar 2019 16:39:00 GMT
View Forum Message <> Reply to Message

Hello,

As of March 18, a new version of libssh2 (1.8.1) is out.
It has many bugfixes and security patches.
I'm currently testing it.
If everything goes well, I will update the SSH package (hopefully befure 2019.1).

Best regards,
Oblivion
