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 apprecieate reviews, bug reports, patches, suggestions, criticism.

Best regards,

Oblivion.

File Attachments

1) SSH-SecondIteration.zip, downloaded 240 times