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
