



Please note that GitHub no longer supports Internet Explorer.

We recommend upgrading to the latest [Microsoft Edge](#), [Google Chrome](#), or [Firefox](#).

[Ignore](#)[Learn more](#)

## Stay up to date on releases

[Dismiss](#)

Create your free account today to subscribe to this repository for notifications about new releases, and build software alongside 36 million developers on GitHub.

[Sign up for free](#)[See pricing for teams and enterprises](#)[Releases](#)[Tags](#)

## 2016.1.0

[Latest release](#)

📦 2016.1.0 🔑 19167e7



drieseng released this Oct 16, 2017 · [33 commits](#) to develop since this release

## Changes

### Distribution

The **NuGet Package Manager** in Visual Studio 2012 does not support packages in which a given dependency is defined for more than one TFM. Since a fix for [this issue](#) is not expected any time soon, we're reviving support for a binary distribution.

This plain .zip file contains the **SSH.NET** assembly, and where applicable any dependencies, for all supported target frameworks.

For Visual Studio 2015 and higher, the **SSH.NET** NuGet package remains our primary distribution mechanism.

### General

- Increase initial window size for SSH channels from **2 MB** to **2147483647 ( $2^{31} - 1$ ) bytes**. This results in less protocol overhead when receiving data from a SSH server.
- Data of an **SSH\_MSG\_IGNORE** server request is now ignored when the specified length is greater than the actual available bytes (issue [#41](#)).
- Reduce overhead when invoking both `Disconnect()` and `Dispose()` .
- Improve performance of SSH message processing by eliminating use of dynamic dispatching.

## ScpClient

A **RemotePathTransformation** property has been added to **ScpClient** that can be used to change if and how a remote path is transformed before it is passed to the **scp** command on the remote server.

**SSH.NET** comes with the following path transformations that are exposed through the **RemotePathTransformation** class (in **Renci.SshNet**):

- **DoubleQuote**  
Encloses a path in double quotes, and escapes any embedded double quote with a backslash.  
This is the **default**, as it works fine with most SCP servers.
- **ShellQuote**  
Quotes a path in a way to be suitable to be used with a shell-based SCP server. This is the preferred mechanism for shell-based SCP servers, which typically means all Unix-like systems.
- **None**  
Performs no transformation to the path.  
This is the recommended transformation when the remote host does not require any quoting to preserve the literal value of metacharacters, or when remote paths are guaranteed not to contain such characters.

When none of the built-in transformations match the specific requirements of a remote host, a [custom transformation](#) can be supplied.

More information on this change is available [here](#).

Fixes issues [#256](#) and [#108](#).

### Example:

Download a file using SCP, and apply shell quoting to the remote path:

```
using Renci.SshNet;

public class Program
{
    static void Main()
    {
        using (var client = new ScpClient("HOST", 22, "USER", "PWD"))
        {
            client.RemotePathTransformation = RemotePathTransformation.ShellQuo
            client.Connect();

            using (var ms = new MemoryStream())
            {
                client.Download("/home/sshnet/file 123", ms);
            }
        }
    }
}
```

Since we've configured **ScpClient** to use the **ShellQuote** transformation, the `/home/sshnet/file 123` path will automatically be enclosed in single quotes.

The actual path that is passed to the **scp** command on the remote host is therefore `'/home/sshnet/file 123'`.

## SftpClient

- When the read buffer is empty, `SftpFileStream.Read(byte[], int, int)` only writes those bytes to the buffer that go beyond the number of bytes requested by the caller.
- Reduced memory allocations in **SftpFileStream** by lazily allocating read and write buffer.
- Improved compatibility of `SftpFileStream.SetLength(long value)` with [System.IO.FileStream](#) (PR #272):
  - Flush buffers before changing the length of stream.

- Move the current position to the last byte of the stream if the current position is greater than the new length.
- Eliminated extra read from server in **SftpFileStream** to determine EOF.
- Greatly improved performance of `SftpClient.(Begin)DownloadFile(...)` by asynchronously reading ahead chunks (issue [#145](#) and [#100](#)).

## SshClient

- A **ForwardedPortDynamic** now supports domain name addresses for SOCKS5 requests (issue [#98](#)).

## Breaking changes

---

### ScpClient

The `Upload(FileInfo fileInfo, String path)` method in **ScpClient** now expects *path* to be the remote path of the file to which you want to upload, and throws a **ScpException** if a directory exists on the remote host for that path (issue [#286](#)).

Up until now, uploading a file with a given (remote) path could lead a different result depending on whether a directory for that path exists on the remote host.

Example:

```
using (var client = new ScpClient("host", "user", "pwd"))
{
    client.Connect();

    client.Upload(new FileInfo(@"c:\temp\newlog.txt", "/home/sshnet/log");
}
```

As of version [2016.1.0-beta 3](#) a **ScpException** will be thrown when `/home/sshnet/log` exists as a directory on the remote host.

In previous versions of **SSH.NET** this code would actually upload the content of the `c:\temp\newlog.txt` file to `/home/sshnet/log/newlog.txt`.

When `/home/sshnet/log` exists on the remote host as a file or does not exist at all, the content of the `c:\temp\newlog.txt` file will be uploaded to `/home/sshnet/log`. This has not changed.

## SftpClient

- The `IsAsync` property was removed from `SftpFileStream`. Previously this property always returned *false*.
- The read and write position in `SftpFileStream` are no longer tracked separately. Reading from or seeking in the `SftpFileStream` will now also affect the position at which a subsequent write is performed, and vice versa (issue [#253](#)).

For example, the following code snippet will now write "123456" to the console:

```
var buffer = Encoding.UTF8.GetBytes("123456");

using (var client = new SftpClient("host", "user", "pwd"))
{
    client.Connect();

    using (var ws = client.OpenWrite(path))
    {
        ws.Write(buffer, 0, 3);
    }

    using (var ws = client.OpenWrite(path))
    {
        ws.Seek(3, SeekOrigin.Begin);
        ws.Write(buffer, 3, 3);
    }

    Console.WriteLine(client.ReadAllText(path, Encoding.UTF8));
}
```

- To improve compatibility of `SftpFileStream` with `System.IO.FileStream`, **Append** mode is now only allowed when combined with write-only access (issue [#267](#)). This only affects `SftpClient.Open(string path, FileMode mode, FileAccess access)`.

The following code snippet will now throw an **ArgumentException**:

```
using (var client = new SftpClient("host", "user", "pwd"))
{
    client.Connect();

    using (var fs = client.Open("/home/user/file.log", FileMode.Append,
    {
    }
    })
    {
    }
```



### Result:

System.ArgumentException: Append mode can be requested only when combined with write-only access.

- To improve compatibility of **SftpClient** with [System.IO.File](#), the following methods now use UTF-8 encoding without a Byte-Order Mark (BOM):
  - void AppendAllLines(string path, IEnumerable<string> contents)
  - void AppendAllText(string path, string contents)
  - StreamWriter AppendText(string path)
  - StreamWriter CreateText(string path)
  - void WriteAllLines(string path, IEnumerable<string> contents)
  - void WriteAllLines(string path, string[] contents)
  - void WriteAllText(string path, string contents)

## Fixes

---

### General

- Servers that do not implement [RFC 4252](#) correctly may lead to stack overflow (issue [#306](#)).

Section 5.1 of the aforementioned RFC states the following on **SSH\_MSG\_USERAUTH\_FAILURE** response:



The value of 'partial success' MUST be TRUE if the authentication request to which this is a response was successful. It MUST be FALSE if the request was not successfully processed.

Some SSH servers set 'partial success' to **TRUE** even if the authentication request was not processed successfully, and do not update the *name-list* to remove the method name that failed.

**SSH.NET** has now been updated only attempt authentication **5** times for a method that was considered partially successful.

If this limit is reached for a given method name, and no other authentication method is available, a **SshAuthenticationException** with message "*Reached authentication attempt limit for method (<method name>)*" will be thrown.

- Key exchange is slow when size of group is more than **1024 bit** (issue [#304](#) and [#130](#)).

As part of the key exchange, **SSH.NET** sends a **SSH\_MSG\_KEY\_DH\_GEX\_REQUEST** with a **1024 bit** minimum and preferred group size, and **8192 bit** maximum group size.

Before this fix, we would generate a private exponent that matches the size of the safe prime generated by the server. In some cases this meant using a **8192 bit** private component to generate the client exchange value, which is a CPU intensive operation.

As from this fix, we generate a private exponent that is twice the hash size with a minimum of **1024 bit**.

- `ConnectSocks5()` throws a **ProxyException** with a wrong message (issue [#167](#)).
- Comments in **ProxyException.cs** are not relevant (issue [#163](#)).
- SSH exception after client connect using .NET 3.5 version (issue [#113](#)).
- Handle leak when connection to SSH server fails (issue [#55](#)).
- Race condition when both server and client close channel (issue [#84](#)).
- Suppressing a not-connected exception on socket shutdown (issue [#86](#)).

- Race condition when both server and client disconnect (issue [#80](#)).
- Do not consume request-specific data for `SSH_MSG_GLOBAL_REQUEST` (issue [#58](#)).
- `SocketAsyncEventArgs` leak establishing socket connection (issue [#133](#) and [#87](#)).

## ScpClient

- `Upload(Stream source, String path)` does not support uploading a file to the home directory using a relative path (issue [#280](#)).

The following code will now just work:

```
using (var client = new ScpClient("host", "user", "pwd"))
{
    client.Connect();

    using (var fs = File.OpenRead(@"c:\file.log"))
    {
        client.Upload(fs, "file.log");
    }
}
```

- `Upload(Stream source, String path)` does not throw an exception when parent directory does not exist (issue [#289](#)).
- `ScpClient` does not support non-ASCII characters in downloaded file names or error message (issue [#281](#)).

## SftpClient

- `DownloadFile (String path, Stream output, Action<UInt64>)` does not perform well on Sun SSH (issue [#292](#)).
- `SftpFileStream` allows invalid combinations of `FileMode` and `FileAccess` (issue [#191](#)).
- `SftpFileStream.ReadByte()` throws `ArgumentException` when no data is available in read buffer (issue [#173](#)).



- **SftpFileStream** also sends **SSH\_FXP\_FSTAT** upon initialization when mode is not **Append** (issue [#154](#)).
- In **Append** mode, **SftpFileStream** throws a **SftpPathNotFoundException** when the specified file does not exist (issue [#266](#)).
- Uploads with a buffer size that is less than or greather than 32 Kb would corrupt (issue [#70](#)).

## SshClient

- `ShellStream.Write(byte[] buffer, Int32 offset, Int32 count)` skips a single byte when buffer is full (PR [#211](#)).
- `CreateShellStream` overloads always use a 1024 byte buffer (issue [#303](#))
- `Dispose()` throws exception when SSH session gets disconnected unexpectedly (issue [#96](#)).
- `ForwardedPort.Stop()` no longer throws **ObjectDisposedException** when the port is disposed.

## ScpClient




- `ScpClient.Upload(DirectoryInfo, string)` adds extra directory level (issue [#128](#)).
- `ScpClient.Upload(DirectoryInfo, string)` applies timestamp of parent directory (issue [#129](#)).

## Release notes

These release notes include all changes since [2016.0.0](#). The individual release notes for the releases leading up to the [2016.1.0](#) release are available here: [beta 1](#), [beta 2](#), [beta 3](#), [beta 4](#).

There have been no changes since [beta 4](#).

### ▼ Assets 4

 <a href="#">SSH.NET-2016.1.0-bin.zip</a>	2.12 MB
 <a href="#">SSH.NET-2016.1.0-help.chm</a>	2.22 MB
 <a href="#">Source code (zip)</a>	



[Source code \(tar.gz\)](#)