Article
Katherine Reid · Apr 24, 2019 5m read

Using and debugging %Net.SSH.Session for SSH connections

The %Net.SSH.Session class lets you connect to servers using SSH. It's most commonly used with SFTP, especially in the FTP inbound and outbound adaptors.

In this article, I'm going to give a quick example of how to connect to an SSH server using the class, describe your options for authenticating, and how to debug when things go wrong.

Here's an example of making the connection:

```
Set SSH = ##class(%Net.SSH.Session).%New()
Set return=SSH.Connect("ftp.intersystems.com")?
```

This creates a new connection, and then connects to the ftp.intersystems.com SFTP server on the default port. At this point, the client and server have picked encryption algorithms and options, but no user has logged in yet.

Once you're connected, you can choose how to authenticate. There are three methods to choose from:

- AuthenticateWithUsername
- AuthenticateWithKeyPair
- AuthenticateWithKeyboardInteractive

Each of these is a different type of authentication. Here's a brief intro to each type:

AuthenticateWithUsername

This uses a username and password.

AuthenticateWithKeyPair

This uses a pair of public and private keys. The public key must have been pre-loaded on the server, and you must have the matching private key. If the private key is encrypted on disk, you should provide a passphrase to decrypt it in the call to the method. Note: you should never send your private key to anyone else.

The public keys should be in OpenSSH format, and the private keys should be PEM encoded. OpenSSH format looks like this:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCfi2Vq+u0rtt2OC84pyrkq1k7WkrS+s76u3a+2gdD43KQ2Z
3vSUUfksymJjp11JBZEpOtBVIAy221UKdc7j7Qk6sUjZaK8LIy+bzDVwMyFWgVvQge7EjdWjrJLBRCDXYML6y
1Y25XexThkTWSGyXzGNdr+wfIHYn/mIt0hfvrusauvT/9Wz8K2MGAj4BL7UQZpFJrlXzGmewe6++6cZDQQYi0
aztwLK798oc9j0LsccdMpqWrjqoU1uANFhYIuUu/T47TEhT+e6M+KFYK5TR998eJTO25IjdN2Tgw0feXhQFF/
nngbol0bA4auSPaZQsgokKK+E+Q/8UtBdetEofuV user@hostname
```

PEM encoded private keys have a header at the top of the file which looks like this:

```
-----BEGIN RSA PRIVATE KEY-----
```

and end with:

```
-----END RSA PRIVATE KEY-----
```

AuthenticateWithKeyboardInteractive

This is a new option available in Cache 2018.1 and later. It lets you perform challenge and response authentication. For example, you might ask for the one-time code sent via text message or generated by a Google authenticator app. To use this form of authentication, you will need to write a lambda function to handle the prompts the server sends.

You may see servers using this with just a username and password prompt in a way that looks identical to password authentication to the user. The SSH debugging flags described below can help you determine if you're seeing this.

A final note on authentication: If you're interested in using two forms of authentication for a single connection, make sure you're using Cache 2018.1 or any version of InterSystems IRIS. There are updates in this version to allow the use of multiple forms, such as keypair and username.

What to do when things go wrong...

Common errors you might see include:

Failed getting banner

This might look like:

```
ERROR #7500: SSH Connect Error '-2146430963': SSH Error [8010100D]: Failed getting ba
nner [FFFFFFFF8010100D] at Session.cpp:231,0
```

Getting the banner is the first thing an SSH client does. If you're seeing this error, you should verify that you're connecting to the right server and that it is an SFTP server.

For example: if the server is actually an FTPS server, you would see this error. FTPS servers use SSL, not SSH, and therefore don't work with the %Net.SSH.Session class. You can use the %Net.FtpSession class to connect to an FTPS server.

Unable to exchange encryption keys

This error might look like:

```
ERROR #7500: SSH Connect Error '-2146430971': SSH Error [80101005]: Unable to exchang
e encryption keys [80101005] at Session.cpp:238,0
```

This error usually means that the client and server couldn't agree on encryption or MAC algorithms. If you see this, you may need to upgrade either the client or server to add support for new algorithms.

If you're using a version of Cache before 2017.1, I would recommend trying 2017.1 or later. The libssh2 library was upgraded in 2017.1 and added multiple new algorithms.

You can find more details in the logs provided by the debugging flags that I describe below.

Invalid signature for supplied public key

```
Error [80101013]: Invalid signature for supplied public key, or bad username/public k
ey combination [80101013] at Session.cpp:418
```

This error can be quite misleading. You'll see this if your server wanted two forms of authentication and you've only provided one. If that's the case, keep going and try the next one! Everything may still work out.

Error -37

You may see messages about error -37. For example, here it is in the debugging log:

```
[libssh2] 0.369332 Failure Event: -37 - Failed getting banner
```

Any time error -37 is listed, the operation which failed will be re-tried. This error is not what caused the final failure. Check for other error messages.

The SSH debuging flags

Detailed logging for SSH connections can be enabled for a connection using the SSH debugging flags. The flags are enabled with the SetTraceMethod method. Here's an example of a connection using them:

```
Set SSH = ##class(%Net.SSH.Session).%New()
Do SSH.SetTraceMask(511,"/tmp/ssh.log")
Set Status=SSH.Connect("ftp.intersystems.com")?
```

The first argument to SetTraceMask tells it what to collect. It is a decimal representation of bits. 511 asks for all the bits except for 512, and is the most commonly used setting. If you'd like to know more about each bit, they are listed in the class documentation for the %Net.SSH.Session class.

The second argument tells it what file to put the logging information about the connection in. In this example, I used the file /tmp/ssh.log, but you can enter any absolute or relative path you want to use.

In the example above, I've only run the Connect method. If your problem is in the authentication, you'll need to run the appropriate authentication method as well.

Once you've run your test, you can check the log file for information. If you're not sure how to interpret the log file, the WRC can help.

#Best Practices #Debugging #FTP #Caché #Ensemble #InterSystems IRIS