asynSSH

EPICS Developers Manual

Issue:	Draft 1.0
Date:	30th May 2013

	NAME	DATE	SIGNATURE
Prepared by	Alan Greer, Observatory Sciences Ltd.	30 May 2013	[illegible]
Checked by	Philip Taylor, Observatory Sciences Ltd.	31 May 2013	[illegible]

TABLE OF CONTENTS

1 Scope 3
2 Reference Documents 3
3 Introduction 4
4 Requirements 4
4.1 Hardware Requirements 4
4.2 Software Requirements 4
5 Installation Under Linux 4
5.1 libssh2 Installation 5
5.2 EPICS Base and Modules 5
5.3 asynSSH Module Installation 5
6 Running the Test Application 7
6.1 Running the EPICS Database 7
6.2 Running the EDM Screens 8
7 Development of the Device Support Code 9
7.1 SSH Driver Class 9
7.2 Asyn SSH Port Driver 11

1 Scope

This document describes the installation and use of the EPICS general purpose SSH asyn driver module. The module provides a standard asyn driver interface for message based communications using the SSH protocol.

2 Reference Documents

[RD1] IOC Application Developer's Guide (EPICS 3.14.12), *Marty Kraimer et al.*
[RD2] EPICS R3.14 Channel Access Reference Manual, *Jeffrey O. Hill.*
[RD3] asynDriver: Asynchronous Driver Support, *Marty Kraimer, Eric Norum and Mark Rivers.*

3 Introduction

Secure Shell (SSH) is a cryptographic network protocol for secure data communication, remote shell services or command execution and other secure network services between two networked computers that connects, via a secure channel over an insecure network, a server and a client.

The asynSSH software module contains a low level SSH wrapper in the form of an asyn driver that can be used to connect to an SSH server. The message based octet interface is implemented providing read/write access.

A small test application is present within the module that opens a connection and performs some simple shell commands using the stream device module. An EDM screen is included that can be used to interact with the test application records.

4 Requirements

This section details the hardware and software requirements for using the asynSSH EPICS module.

4.1 Hardware Requirements

The following hardware is required to run the module code:

- Standard PC, with an Ethernet port if connecting to a server across a network. The module can be tested making a connection to localhost if required.

4.2 Software Requirements

The following software must be installed to run the driver and device support code:

- Linux operating system. The module has been written using general asyn and EPICS base function calls and methods and as a result will be fully functional on any system that can compile the required versions of EPICS base and the asynDriver. In addition the libssh2 library is required, and although it is possible to compile that library for Windows this has not been tested with this module. The code was developed on CentOS 5.7 (32 bit).
- EPICS Base (version 3.14.12 or later)
- EPICS module 'AsynDriver' (version 4.17 or later)
- EPICS module 'StreamDevice' (version 2.5 or later) required for the test application.
- EDM to run test application screen, if it is required.
- libssh2 library (not an EPICS module, see installation instructions below).

5 Installation Under Linux

This section covers installation of the asynSSH module on a Linux operating system.

5.1 libssh2 Installation

The lower level driver class provided by the EPICS module requires that an external library be installed on the system to handle SSH encryption and connections. The library libssh2 (http://www.libssh2.org) should be downloaded and installed. The version that the driver has been built against is 2-1.4.3.

Either download using the link on the libssh2 website, or from a command line download using wget:

```
wget http://www.libssh2.org/download/libssh2-1.4.3.tar.gz
```

Unpack the archive to a suitable location, configure, build and install.

```
gunzip libssh2-1.4.3.tar.gz
tar -xvf libssh2-1.4.3.tar
cd libssh2-1.4.3
./configure
make
su
make install
```

The libraries are by default installed in /usr/local/lib. Note that installation in that location requires root access.

Note that on many operating systems libssh2 packages already exists but the current package version is not yet compatible with this module.

5.2 EPICS Base and Modules

The asynSSH module has been developed against EPICS base version 3.14.12.1 and asynDriver version 4.17. These modules must be installed before attempting to compile the asynSSH module. If the test application is to be built then an additional module, StreamDevice version 2.5 is also required.

EPICS base can be downloaded from http://www.aps.anl.gov/epics.

The asynDriver EPICS module can be downloaded from http://www.aps.anl.gov/epics/modules/soft/asyn and installation instructions can be found on the website, they are beyond the scope of this document.

The Stream Device module can be downloaded from http://epics.web.psi.ch/software/streamdevice and installation instructions can be found on the website, they are beyond the scope of this document.

5.3 asynSSH Module Installation

Assuming the required software described above has been installed, unpack the asynSSH EPICS module archive file and cd into the top directory.

```
gunzip asynSSH_V0-1.tar.gz
tar -xvf asynSSH_V0-1.tar
```

```
cd asynSSH_V0-1
```

In here you will see the following files and directories:

Directory	Description
asynSSHApp	This directory contains the main driver source code.
asynSSHTestApp	This directory contains a test application. A database file sets up a system with a few test records for various shell commands. An edm screen is provided to interact with the test application.
configure	The configuration directory.
iocBoot	The boot directory for the test application. It contains the startup script that defines the address of the SSH server as well as the username and password. These must be changed for the connection to be successful.
Makefile	Top-level Make file to build all the software

Before building the application it is necessary to define the installation locations of the EPICS installation, asyn and stream modules, and a link to the libssh2 library. cd into the configure directory and edit the RELEASE file. Update the lines that define the location of the asyn and stream installations to point to wherever these modules are installed on your system. For example

```
ASYN=/usr/lib/epics/asyn-4.17
STREAM=/usr/lib/epics/stream2-5
```

Also update the line that defines the location of your EPICS installation. For example

```
EPICS_BASE=/usr/lib/epics
```

Save any changes and exit. cd back up to the top level and then into the application source directory.

```
cd ..
cd asynSSHApp/src
```

It is also necessary to edit the Makefile in that source directory to ensure it points to the correct installation location of libssh2. In the Makefile there is a variable ssh2_DIR that should be set to the location of the installed library file. For example

```
ssh2_DIR = /usr/local/lib
```

Finally the host architecture must be defined before the build can commence. From a terminal (assumed to be running the Bash shell) enter the following line:

```
export EPICS_HOST_ARCH=linux-x86
```

There is a test application supplied with the device support code. If this is to be used then the startup script should be altered. cd into the iocBoot/iocasynSSHTest directory from the top level.

```
cd iocBoot/iocasynSSHTest
```

Edit the st.cmd file and update the line that configures the asyn SSH port with the IP address of the SSH server on your network. Specify the username for that server and also a password if required. The driver will attempt a form of key based authorization if no password is supplied. For example

```
drvAsynSSHPortConfigure("SSH1","localhost","username",
"password","0","0","0")
```

Now cd back to the top level and type ‘make’ to build the device support code and the test application. The build should complete with no errors.

6 Running the Test Application

Once the system has successfully compiled the test application should be executed to ensure a connection to the SSH server is completed.

6.1 Running the EPICS Database

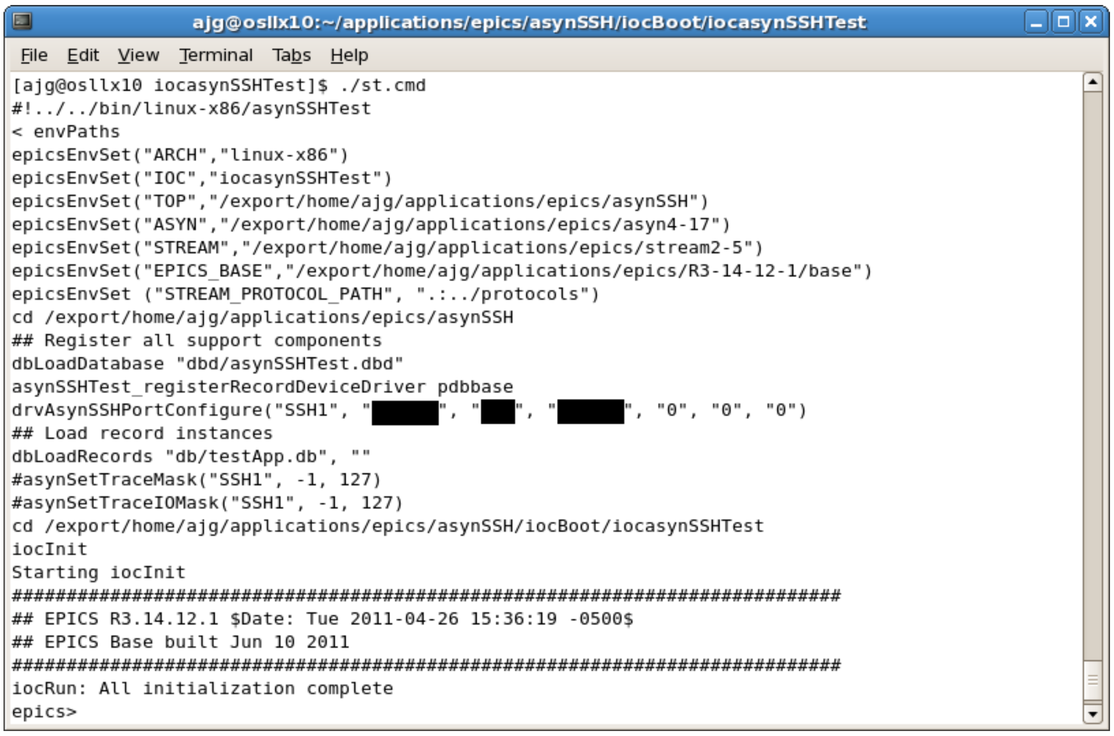
To run the IOC cd into the iocBoot/iocasynSSHTest directory from the top level directory and execute the st.cmd script.

```
cd iocBoot/iocasynSSHTest
./st.cmd
```

It may be necessary to change the permissions of the bash script to make it executable. Use a chmod command from the command line to achieve this.

```
chmod 755 st.cmd
```

Below is an example of the output generated when the system is started on CentOS.


ajg@osllx10:~/applications/epics/asynSSH/iocBoot/iocasynSSHTest
File Edit View Terminal Tabs Help

```
[ajg@osllx10 iocasynSSHTest]$ ./st.cmd
#!../../bin/linux-x86/asynSSHTest
< envPaths
epicsEnvSet("ARCH","linux-x86")
epicsEnvSet("IOC","iocasynSSHTest")
epicsEnvSet("TOP","/export/home/ajg/applications/epics/asynSSH")
epicsEnvSet("ASYN","/export/home/ajg/applications/epics/asyn4-17")
epicsEnvSet("STREAM","/export/home/ajg/applications/epics/stream2-5")
epicsEnvSet("EPICS_BASE","/export/home/ajg/applications/epics/R3-14-12-1/base")
epicsEnvSet ("STREAM_PROTOCOL_PATH", ".:../protocols")
cd /export/home/ajg/applications/epics/asynSSH
## Register all support components
dbLoadDatabase "dbd/asynSSHTest.dbd"
asynSSHTest_registerRecordDeviceDriver pdbbase
drvAsynSSHPortConfigure("SSH1", "[redacted]", "[redacted]", "[redacted]", "0", "0", "0")
## Load record instances
dbLoadRecords "db/testApp.db", ""
#asynSetTraceMask("SSH1", -1, 127)
#asynSetTraceIOMask("SSH1", -1, 127)
cd /export/home/ajg/applications/epics/asynSSH/iocBoot/iocasynSSHTest
iocInit
Starting iocInit
############################################################################
## EPICS R3.14.12.1 $Date: Tue 2011-04-26 15:36:19 -0500$
## EPICS Base built Jun 10 2011
############################################################################
iocRun: All initialization complete
epics>
```

Figure 1 IOC output during start-up.

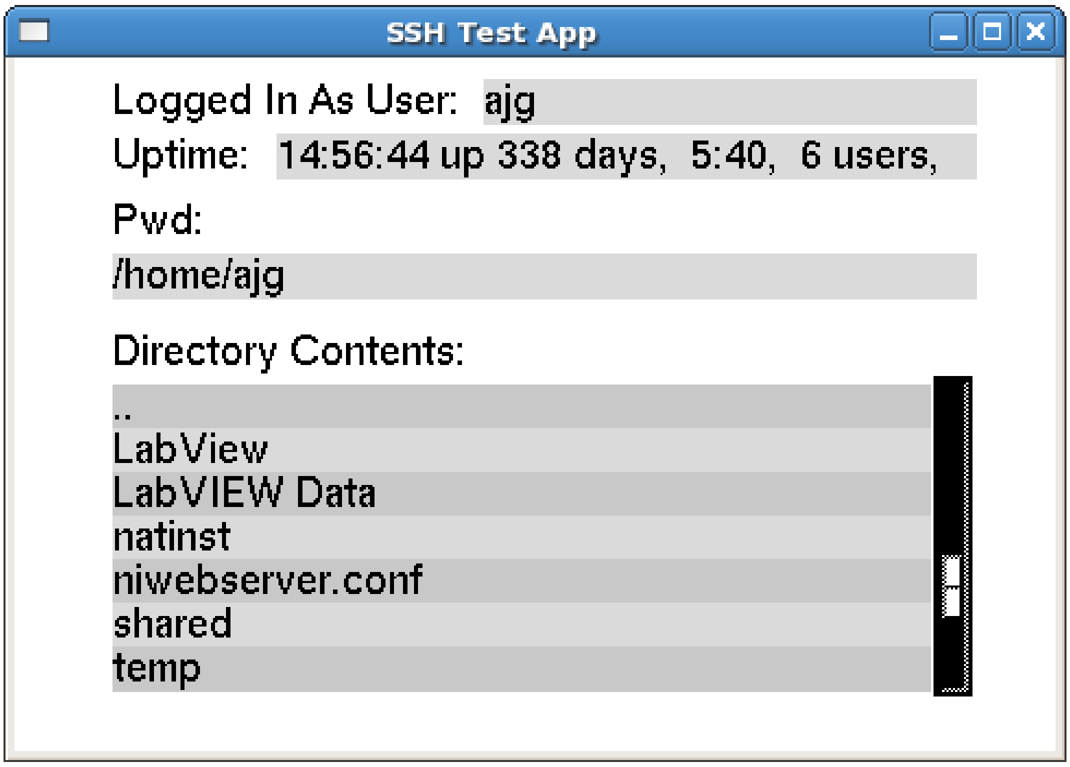
6.2 Running the EDM Screens

To allow easy testing, some EDM engineering screens have been added; these can be started on Linux by changing directory to asynSSHTestApp/edl and executing edm

```
edm -x sshTest.edl
```

The screens can be converted to other formats (http://www.aps.anl.gov/epics/) if required.

Once started the user is presented with the test screen. The test application contains some records to provide some information on the SSH server as well as the current directory location and contents. Clicking on any of the folders displayed will execute a cd command on the server to change to that directory. If the entry clicked on is not a directory then the location will not update.



Logged In As User: ajg
Uptime: 14:56:44 up 338 days, 5:40, 6 users,
Pwd:
/home/ajg
Directory Contents:

```
..
LabView
LabVIEW Data
natinst
niwebserver.conf
shared
temp
```

Figure 2 Main EDM screen.

7 Development of the Device Support Code

This section describes in more detail the main files/classes involved in the asynSSH EPICS module. These files can be located in the source directory of the asynSSH module.

7.1 SSH Driver Class

Filenames	sshDriver.h sshDriver.cpp
Location	${TOP}/asynSSHApp/src
Methods	SSHDriver setUsername setPassword connectSSH setBlocking flush write read disconnectSSH ~SSHDriver

The SSH driver class provides a wrapper around the libssh2 library and presents a simple interface for establishing connections to the SSH server and write/read/flush operations.

Method	SSHDriver
Parameters	Host - Host name or IP of device.
Return	N/A
Description	Constructor for the driver class. Initializes internal variables.

Method	setUsername
Parameters	Username - The username for the SSH connection.
Return	SSHDriverStatus.
Description	Setter for the username.

Method	setPassword
Parameters	Password - The password for the SSH connection.
Return	SSHDriverStatus.
Description	Setter for the password. If this method is not called then authentication using keys is attempted when connecting.

Method	connectSSH
Parameters	None.
Return	SSHDriverStatus.
Description	Attempt to create a connection and authorize the username with the password or by keys. Once a connection has been established a dumb terminal is created.

Method	setBlocking
Parameters	Blocking - 0 for non-blocking, 1 for blocking.
Return	SSHDriverStatus.
Description	This is a private method and will always be set to 0 for the asynSSH module to avoid blocking read calls.

Method	flush
Parameters	None.
Return	SSHDriverStatus.
Description	Attempt to flush the connection. Also performs a read to ensure no bytes are hanging around for the next read.

Method	write
Parameters	Buffer - The ascii buffer to write. BufferSize - The number of bytes to write (size of buffer). BytesWritten - The actual number of bytes that were written. Timeout - The number of milliseconds to wait before timing out.
Return	SSHDriverStatus.
Description	Write a message to the connection. The echoed message is then read back from the connection so that the next read does not contain these extra bytes.

Method	read
Parameters	Buffer - A string buffer to hold the data that has been read. BufferSize - The maximum number of bytes to read (the size of the buffer).

	BytesRead - The actual number of bytes read. Timeout - The number of milliseconds to wait before timing out.
Return	SSHDriverStatus.
Description	Read data from the connected channel. The read method will continue to attempt to read data from the channel until either data arrives or the timeout is reached.

Method	disconnectSSH
Parameters	None.
Return	SSHDriverStatus.
Description	Close the connection.

Method	~SSHDriver
Parameters	None.
Return	N/A
Description	Destructor.

7.2 Asyn SSH Port Driver

Filenames	drvAsynSSHPort.h drvAsynSSHPort.cpp drvAsynSSHPort.dbd
Location	${TOP}/asynSSHApp/src
Functions	closeConnection asynCommonReport cleanup connectIt asynCommonConnect asynCommonDisconnect writeIt readIt flushIt sshCleanup drvAsynSSHPortConfigure

These functions implement an Asyn Octet interface, allowing higher level record device support to connect, lock and communicate with the driver and the SSH server. This driver code also handles loss of connection and reconnection, keeping the interface clean for the higher level.

Function	closeConnection
Parameters	pasynUser - Pointer to asyn user structure. ssh - Pointer to SSH controller structure. why - String reason for connection close.
Return	void
Description	Closes the connection. Calls disconnect on the driver class if the object exists. Then destroys and cleans up the driver.

Function	asynCommonReport
Parameters	drvPvt - Void pointer to the SSH controller structure. fp - Pointer to a FILE structure. details - The level of detail to provide in the report.
Return	void
Description	Prints a report to the file specified.

Function	cleanup
Parameters	arg - Void pointer to the SSH controller structure.
Return	void
Description	Locks the asyn port, then disconnects and destroys the driver.

Function	connectIt
Parameters	drvPvt - Void pointer to the SSH controller structure. pasynUser - Pointer to asyn user structure.
Return	asynStatus
Description	Creates a new SSH driver object. Sets the username and password according to supplied parameters in the SSH controller structure and then attempts a connection to the server.

Function	asynCommonConnect
Parameters	drvPvt - Void pointer to the SSH controller structure. pasynUser - Pointer to asyn user structure.
Return	asynStatus
Description	Calls connectIt.

Function	asynCommonDisconnect
Parameters	drvPvt - Void pointer to the SSH controller structure. pasynUser - Pointer to asyn user structure.
Return	asynStatus
Description	Calls closeConnection.

Function	writeIt
Parameters	drvPvt - Void pointer to the SSH controller structure. pasynUser - Pointer to asyn user structure. data - ASCII string to write. numchars - Number of characters to write. nbytesTransferred - Number of characters (bytes) that have actually been written.
Return	asynStatus
Description	First checks if a connection exists. If not it calls connectIt. Then a call to the SSH driver write method is made.

Function	readIt
Parameters	drvPvt - Void pointer to the SSH controller structure. pasynUser - Pointer to asyn user structure. data - Buffer ready to accept the response string. maxchars - The maximum size of data (buffer size) that can be received.

	nbytesTransfered - Number of bytes that have actually been read. gotEom - Was an end of message character returned?
Return	asynStatus
Description	First checks if a connection exists. If not it calls connectIt. Then a call to the SSH driver read method is made and the returned bytes placed into the data buffer.

Function	flushIt
Parameters	drvPvt - Void pointer to the SSH controller structure. pasynUser - Pointer to asyn user structure.
Return	asynStatus
Description	Calls the flush method on the SSH driver.

Function	sshCleanup
Parameters	ssh - Pointer to the SSH controller structure.
Return	void
Description	Frees all memory allocated to the SSH controller structure.

Function	drvAsynSSHPortConfigure
Parameters	portName - Name of the asyn port assigned to this driver. hostName - Name (or IP) of the server to connect to. userName - Username to use when establishing an SSH connection. password - Password to use when establishing an SSH connection. priority - The priority of the driver. 0 indicates epicsThreadPriorityMedium. noAutoConnect - 0 indicates asyn automatically connects the port. noProcessEos - 0.
Return	int
Description	This function is (indirectly) called from the EPICS IOC startup script entry, and allocates the SSH controller structure. It then populates the structure with the necessary information and registers the relevant functions with the asyn interface layers. Functions are registered with the common interface and the octet interface.