# How to Demonstrate Junos Pulse L3 VPN - Android

# Setup and Demonstration

## Demonstrating Junos Pulse L3 VPN on Android

In order to demonstrate VPN capabilities in the latest release of Junos Pulse three things are required:

1. The Android Debug Bridge - part of the Android SDK.
2. A working USB connection to the Droid2.
3. Junos Pulse for Android, version 2.1.2.11633.

Android Debug Bridge (adb) is a command line tool that lets you communicate with a connected Android-powered device. It is a client-server program that includes three components:

A client, which runs on your PC. You can invoke a client from a shell by issuing an adb command.

A server, which runs as a background process on your PC. The server manages communication between the client and the adb daemon running on a device.

A daemon, which runs as a background process on a device.

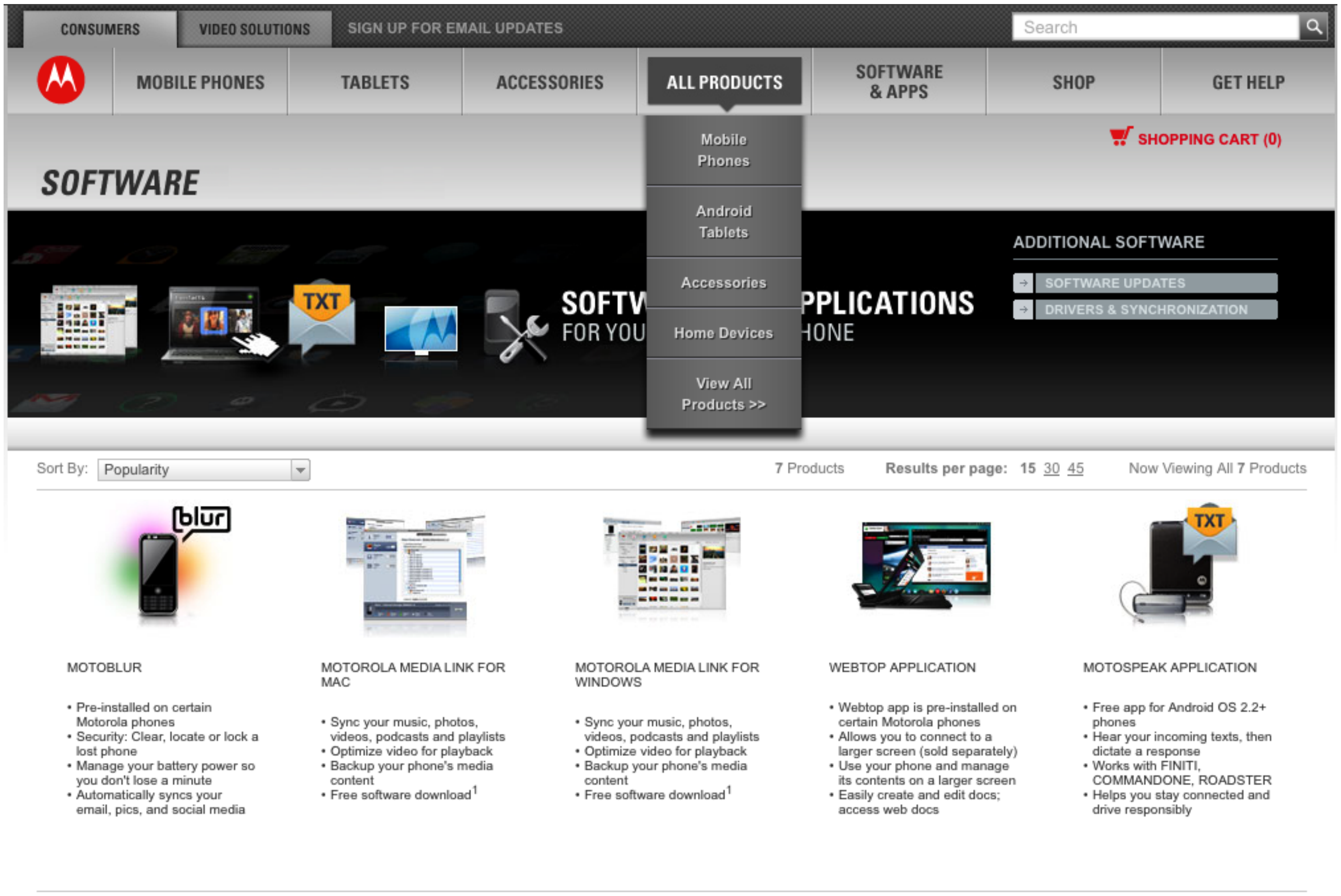
**You can find the adb tool in <sdk>/platform-tools/.**

When you start an adb client, the client first checks whether there is an adb server process already running. If there isn't, it starts the server process. When the server starts, it binds to local TCP port 5037 and listens for commands sent from adb clients—all adb clients use port 5037 to communicate with the adb server.

****Disclaimer** I do not accept responsibility for rooting and bricking your phone by issuing the wrong commands.**

**This document assumes that your device is rooted** - rooting is easy to do but things can happen that potentially could damage the ROM. **Z4ROOT**, a one step application, was used to root the subject Droid2.

## Download and Install MediaLink and the Motorola USB drivers for PC



MOTOBLUR

- Pre-installed on certain Motorola phones
- Security: Clear, locate or lock a lost phone
- Manage your battery power so you don't lose a minute
- Automatically syncs your email, pics, and social media

MOTOROLA MEDIA LINK FOR MAC

- Sync your music, photos, videos, podcasts and playlists
- Optimize video for playback
- Backup your phone's media content
- Free software download[1]

MOTOROLA MEDIA LINK FOR WINDOWS

- Sync your music, photos, videos, podcasts and playlists
- Optimize video for playback
- Backup your phone's media content
- Free software download[1]

WEBTOP APPLICATION

- Webtop app is pre-installed on certain Motorola phones
- Allows you to connect to a larger screen (sold separately)
- Use your phone and manage its contents on a larger screen
- Easily create and edit docs; access web docs

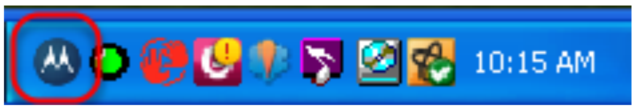
MOTOSPEAK APPLICATION

- Free app for Android OS 2.2+ phones
- Hear your incoming texts, then dictate a response
- Works with FINITI, COMMANDONE, ROADSTER
- Helps you stay connected and drive responsibly

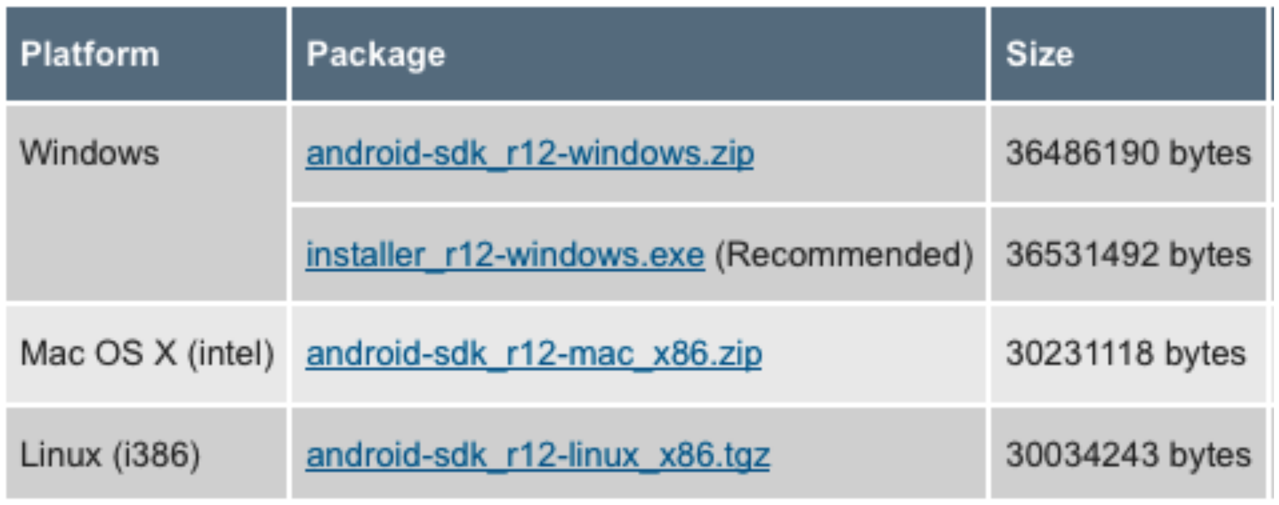
Open your browser and navigate to the Motorola web site.

http://www.motorola.com/Consumers/US-EN/Consumer-Product-and-Services/Software

Select Motorola MediaLink for Windows and install the application. This will also install the Motorola USB drivers on your PC.



## Download and Install the Android SDK



| Platform | Package | Size |
|---|---|---|
| Windows | android-sdk_r12-windows.zip | 36486190 bytes |
| | installer_r12-windows.exe (Recommended) | 36531492 bytes |
| Mac OS X (intel) | android-sdk_r12-mac_x86.zip | 30231118 bytes |
| Linux (i386) | android-sdk_r12-linux_x86.tgz | 30034243 bytes |

Open you browser and navigate to the Android Developers site:

http://developer.android.com/sdk/index.html

Here's an overview of the steps you must follow to set up the Android SDK:

1. Prepare your development computer and ensure it meets the system requirements.

2. Install the SDK starter package from the table above. (If you're on Windows, download the installer for help with the initial setup.)

To get started, download the appropriate package from the table above, then read the guide to Installing the SDK.

3. Make a folder on the root of your 'c' drive named android (mine is c:\androidSDK and is the path Ill use in this guide).

4. Extract the contents of the zip file then navigate through the extracted file 1 level until you see the following files: add-ons, platforms, tools, SDK readme, and SDK setup.

5. Copy all those files into the c:\androidsdk folder.

**UPDATE:** If it says adb not found or something similar, download the ADB files and extract them into the Tools folder of the SDK (they were taken out by Google in the R08 version for some reason and need to be put back).

## Download and Install the tun.ko Tunnel Interface Driver

Search the internet for the tun.ko driver that is known to work. (Droid2 tun.ko file is available at: https://matrix.juniper.net/docs/DOC-79083)

Copy this file to the root of the SDCARD of the device.

**See the Appendix for how you can do this easilly using the adb.

## Connecting to and Testing the ADB Shell Connection

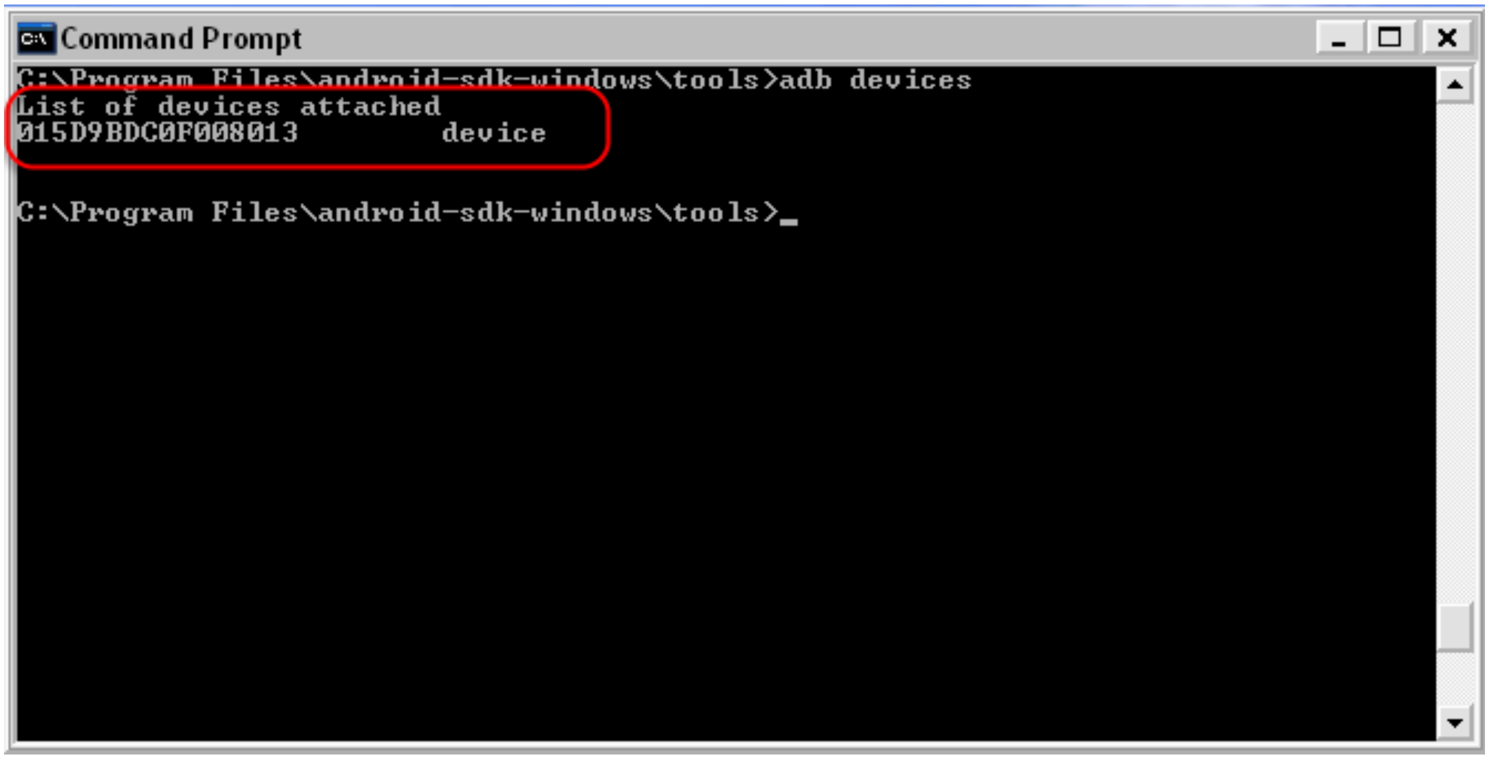
On your Droid2 navigate to: **settings>applications>developement** and select **usb debugging** as this allows for the shell to interface with your phone.

Plug in your Droid2 with the usb cable provided with your phone and wait for drivers to load.

Then on your computer hit start and type "cmd" (without quotes) in the search box (vista/7) or select run and type cmd (XP) then hit enter. You now have a virtual MS DOS window open type the following (each line is its own command and hit enter after typing each line):

```
> cd\
> cd androidsdk\tools
> adb devices
```

This should come up with your devices serial number and show that the connection is ready to be initiated.



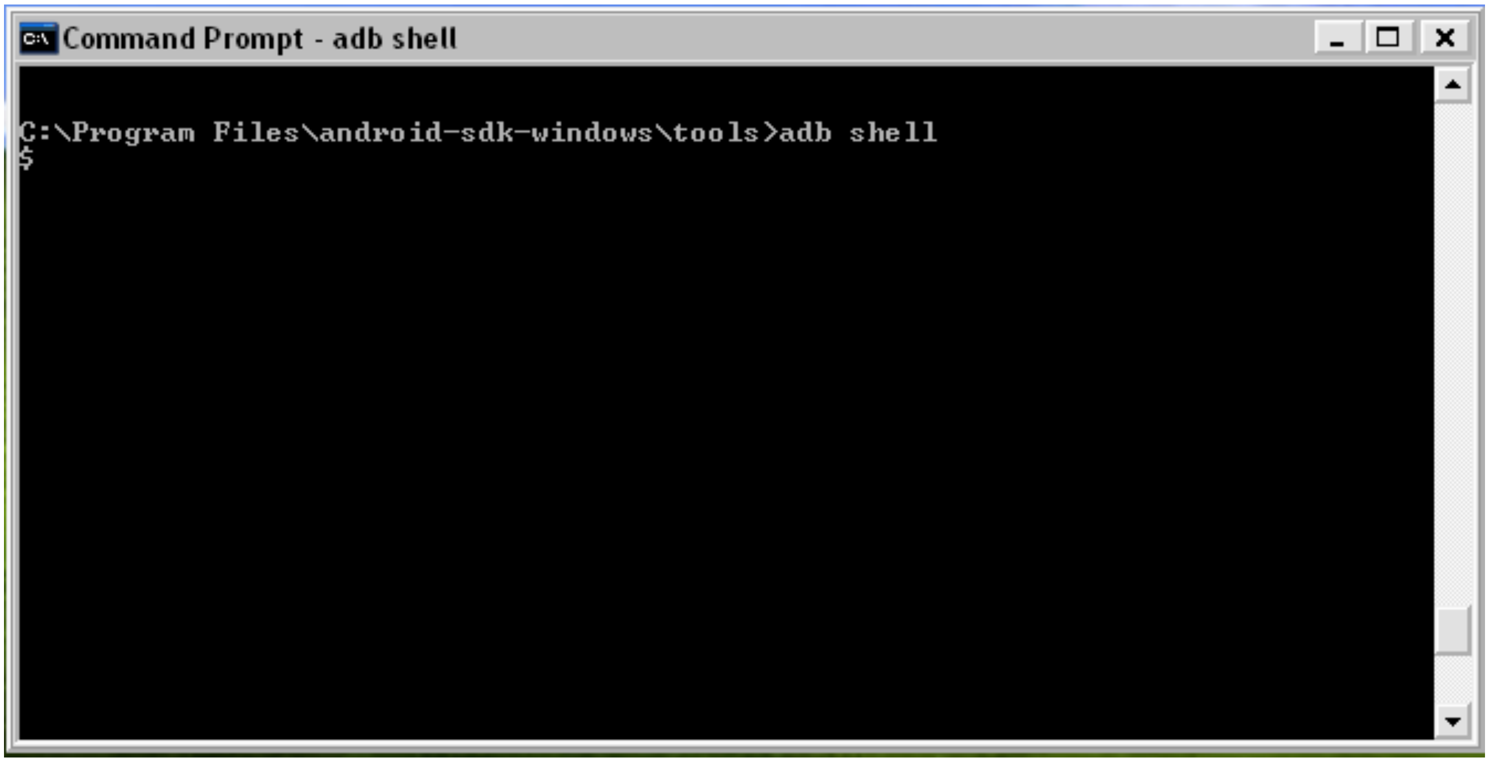
```
C:\Program Files\android-sdk-windows\tools>adb devices
List of devices attached
015D9BDC0F008013        device

C:\Program Files\android-sdk-windows\tools>_
```

## Running the ADB Shell

Type the following to enter the ADB shell (each line is its own command and hit enter after typing each line):

> **adb shell**

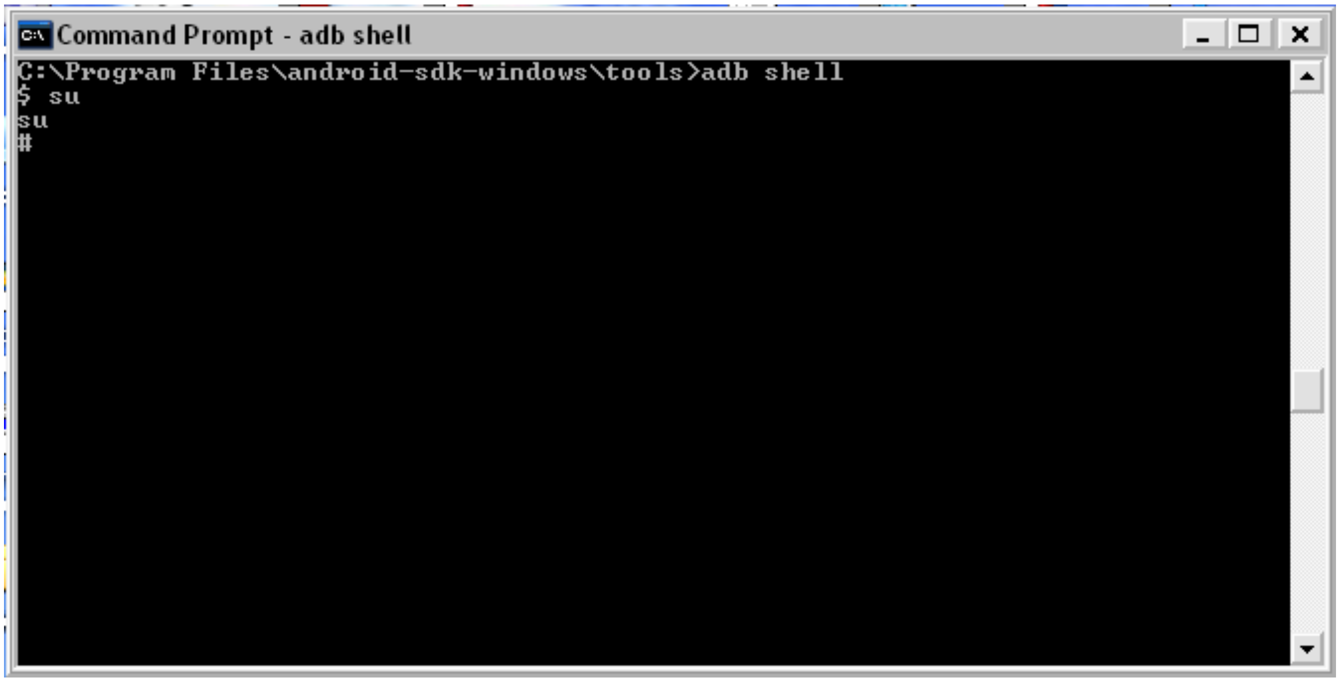
You should now see a "$" (no quotes) and that signifies that you have the adb shell up and running. You're now able to issue commands to your phone over your PC.



Command Prompt - adb shell

```
C:\Program Files\android-sdk-windows\tools>adb shell
$
```

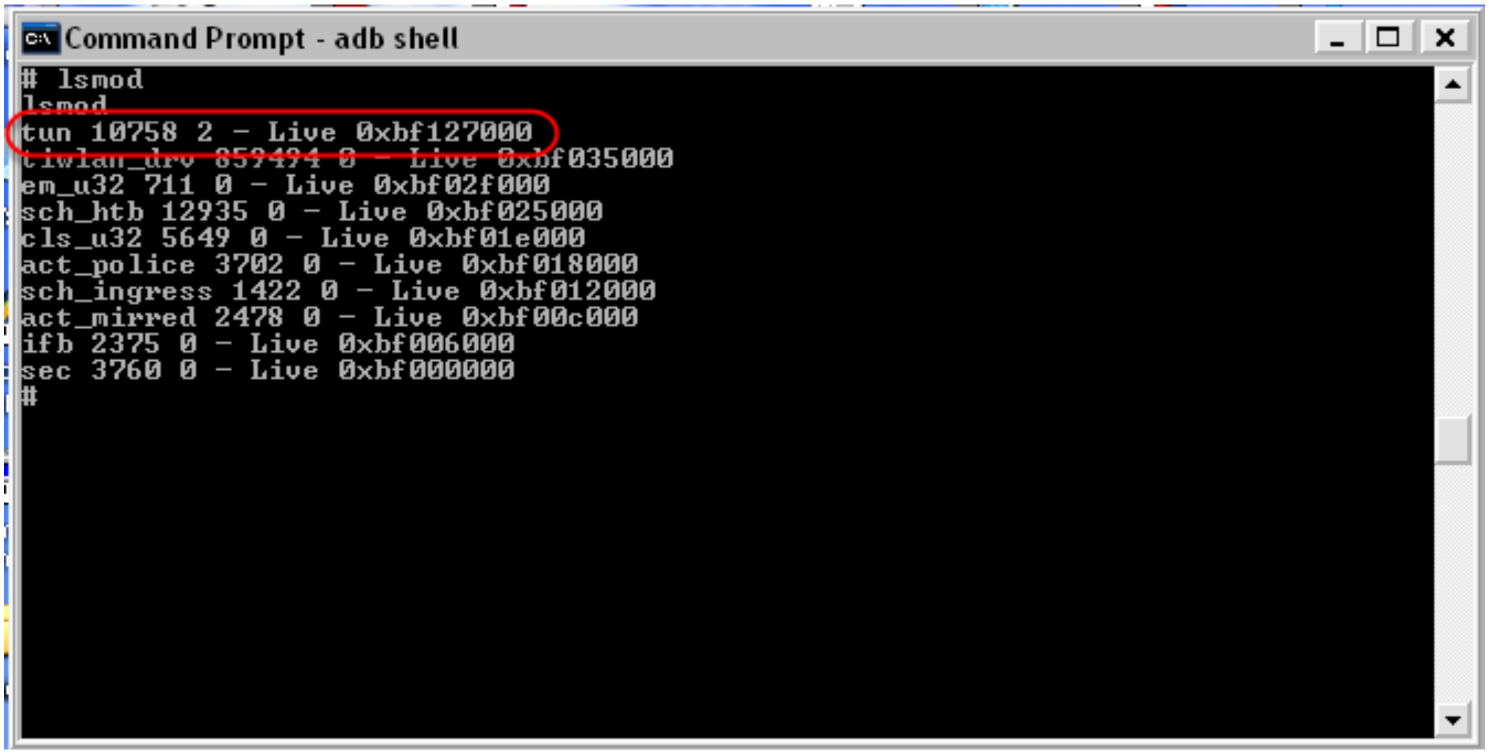
Type "id' to see if you are in superuser mode.  If not, type "su" to enter superuser mode.  You should get a "#" prompt.



Command Prompt - adb shell

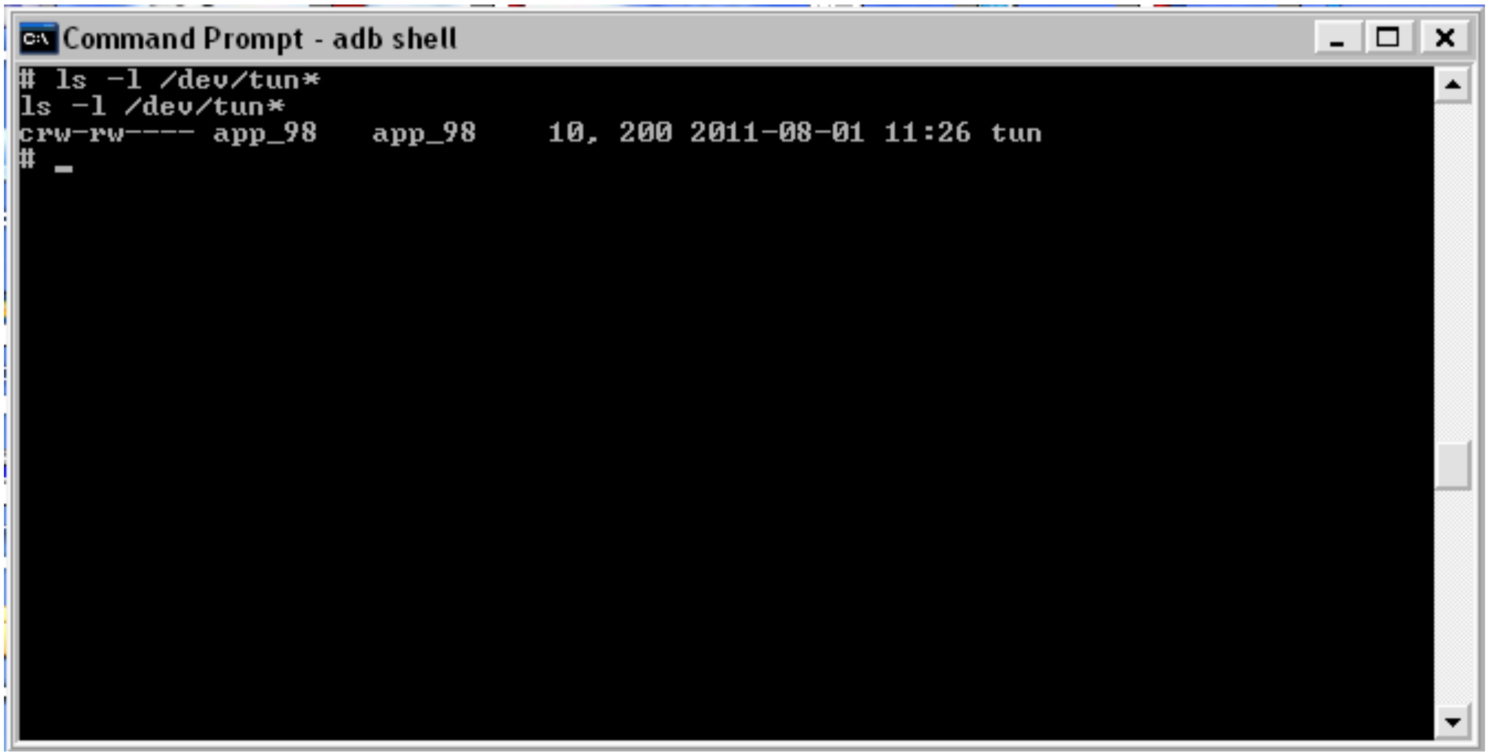
```
C:\Program Files\android-sdk-windows\tools>adb shell
$ su
su
#
```

Now, type "lsmod" to see if you have a tun interface.  # **lsmod**



```
# lsmod
lsmod
tun 10758 2 - Live 0xbf127000
tiwlan_drv 859494 0 - Live 0xbf035000
em_u32 711 0 - Live 0xbf02f000
sch_htb 12935 0 - Live 0xbf025000
cls_u32 5649 0 - Live 0xbf01e000
act_police 3702 0 - Live 0xbf018000
sch_ingress 1422 0 - Live 0xbf012000
act_mirred 2478 0 - Live 0xbf00c000
ifb 2375 0 - Live 0xbf006000
sec 3760 0 - Live 0xbf000000
#
```

Another way to see this interface is to issue the "ls –l /dev/tun*" command # **ls –l /dev/tun***

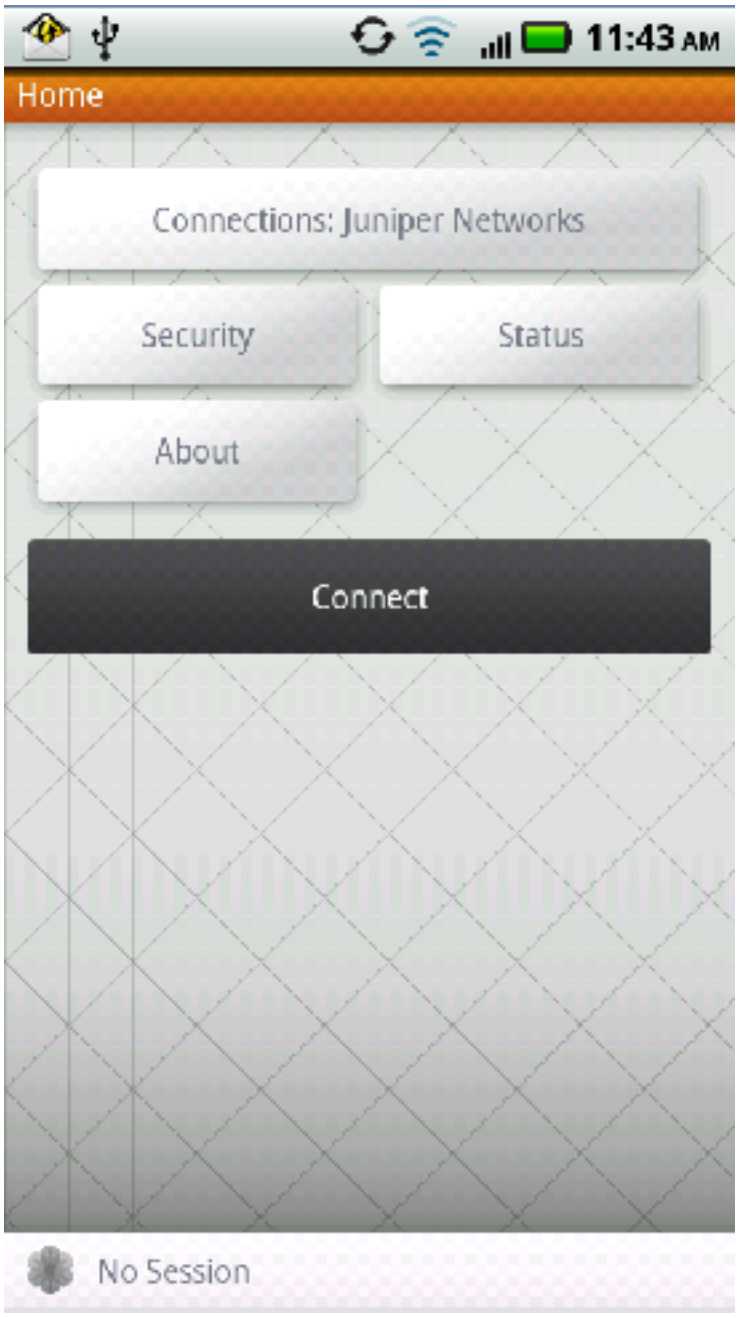


```
# ls -l /dev/tun*
ls -l /dev/tun*
crw-rw---- app_98   app_98    10, 200 2011-08-01 11:26 tun
# _
```

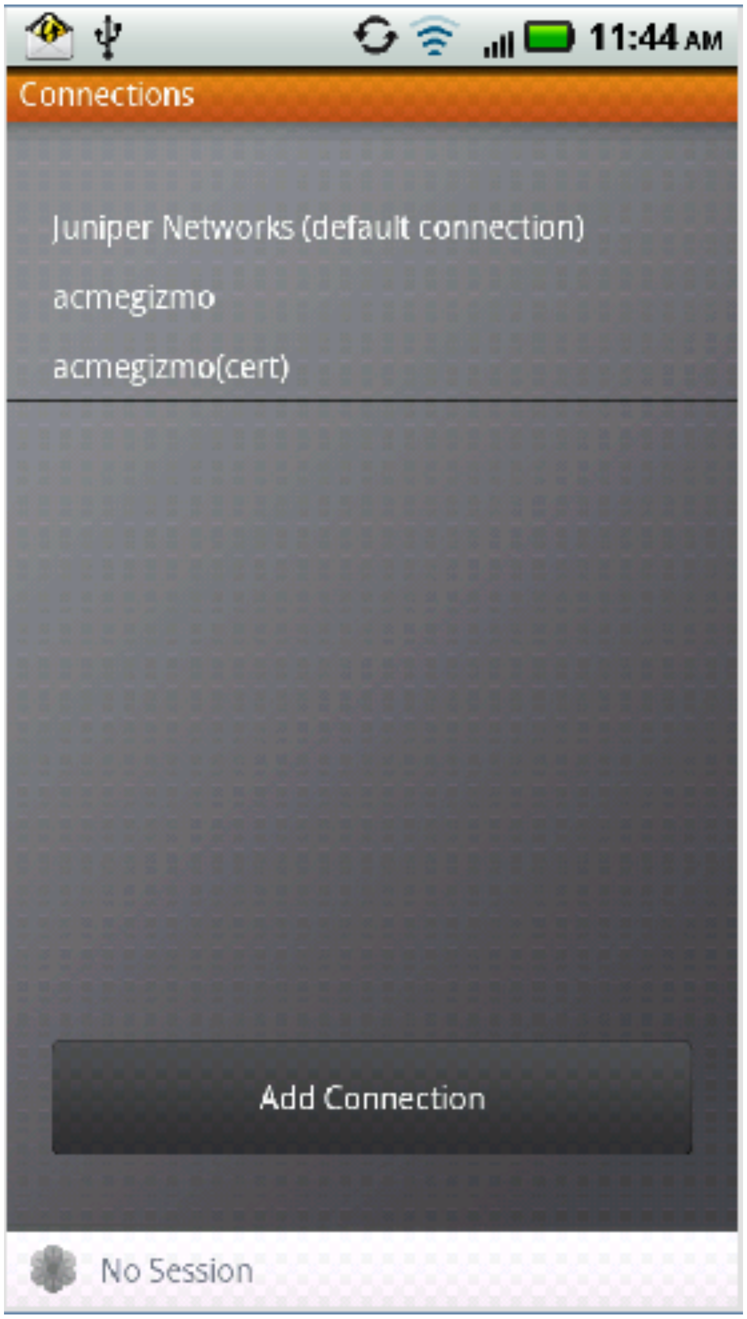
## Install Junos Pulse for Android from the Market

If Junos Pulse is already installed, uninstall it.

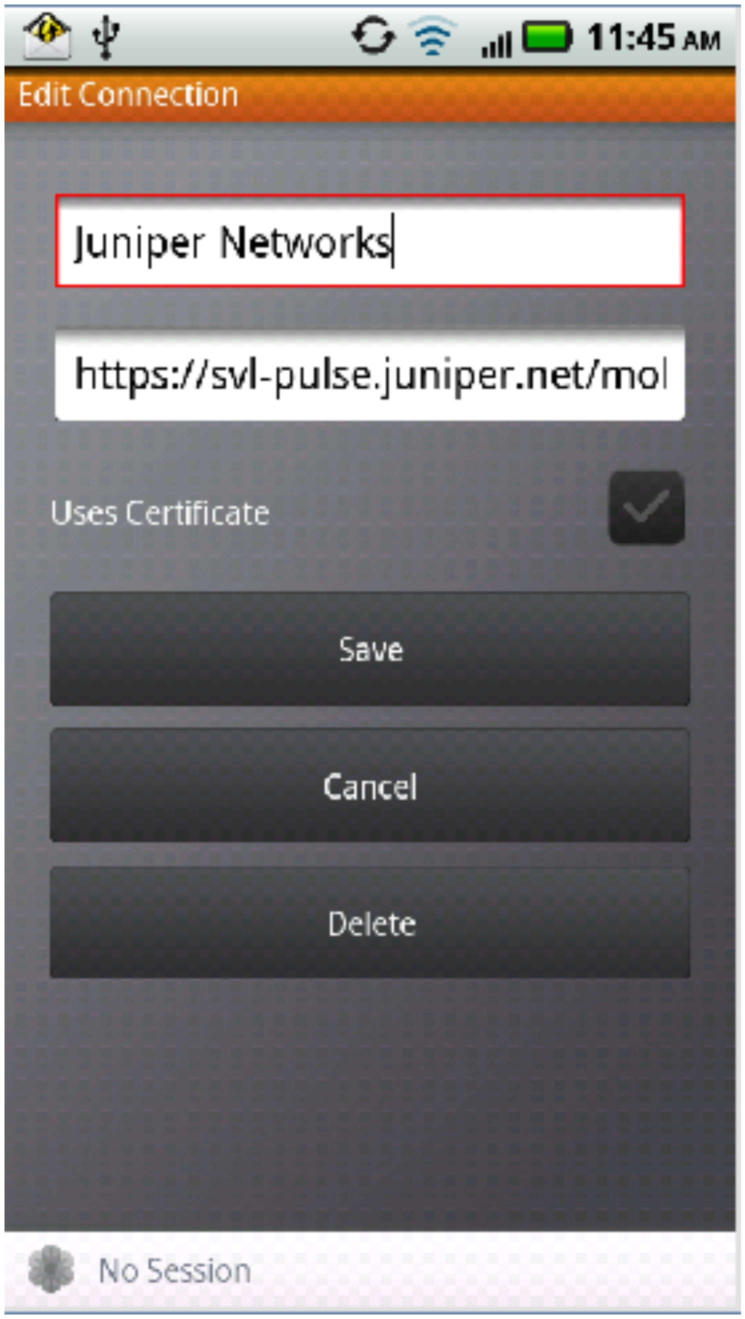
Navigate to the Android Market and search for Junos Pulse, install the allplication accepting permission requests.  Launce Junos Pulse.



Select **Connections:**



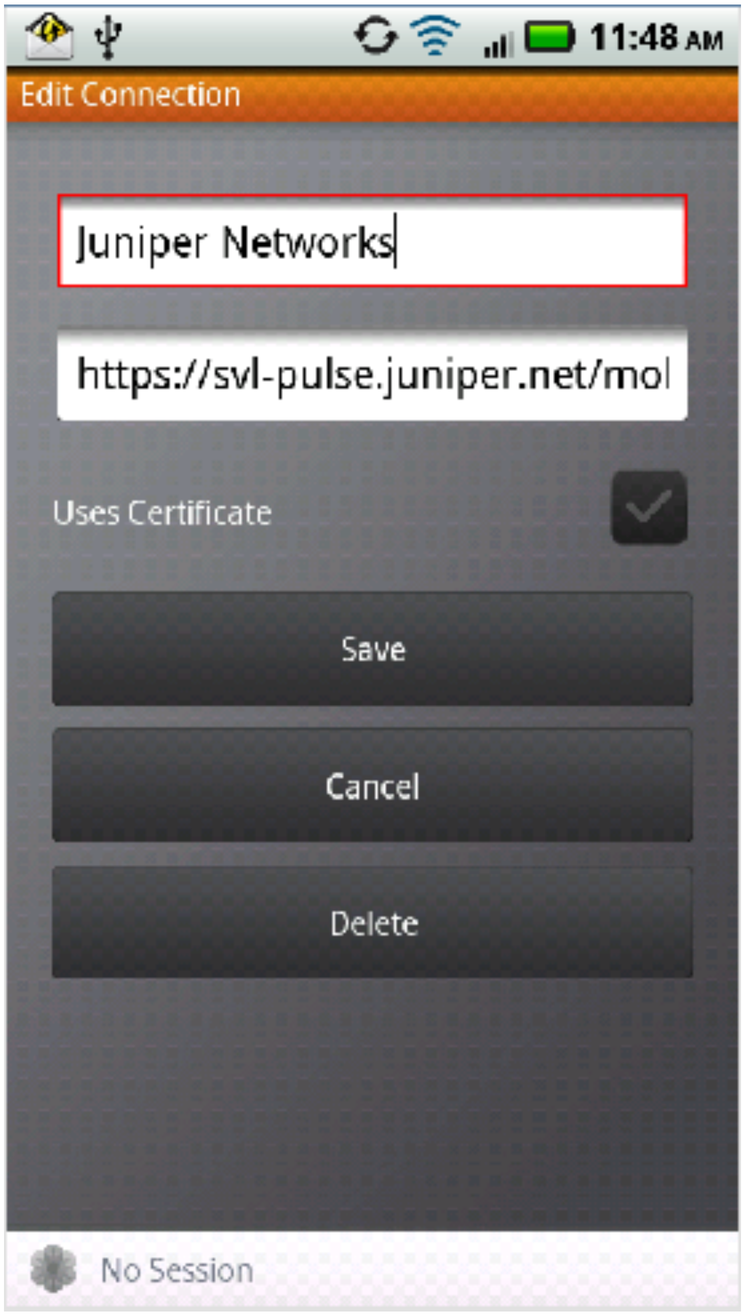
Select **Add Connection**



Give the connection a name  Example: **Juniper Networks Intranet**

Input the URL of the SA/MAG:  Example: **svl-pulse.juniper.net/mobile**

Select **Save**

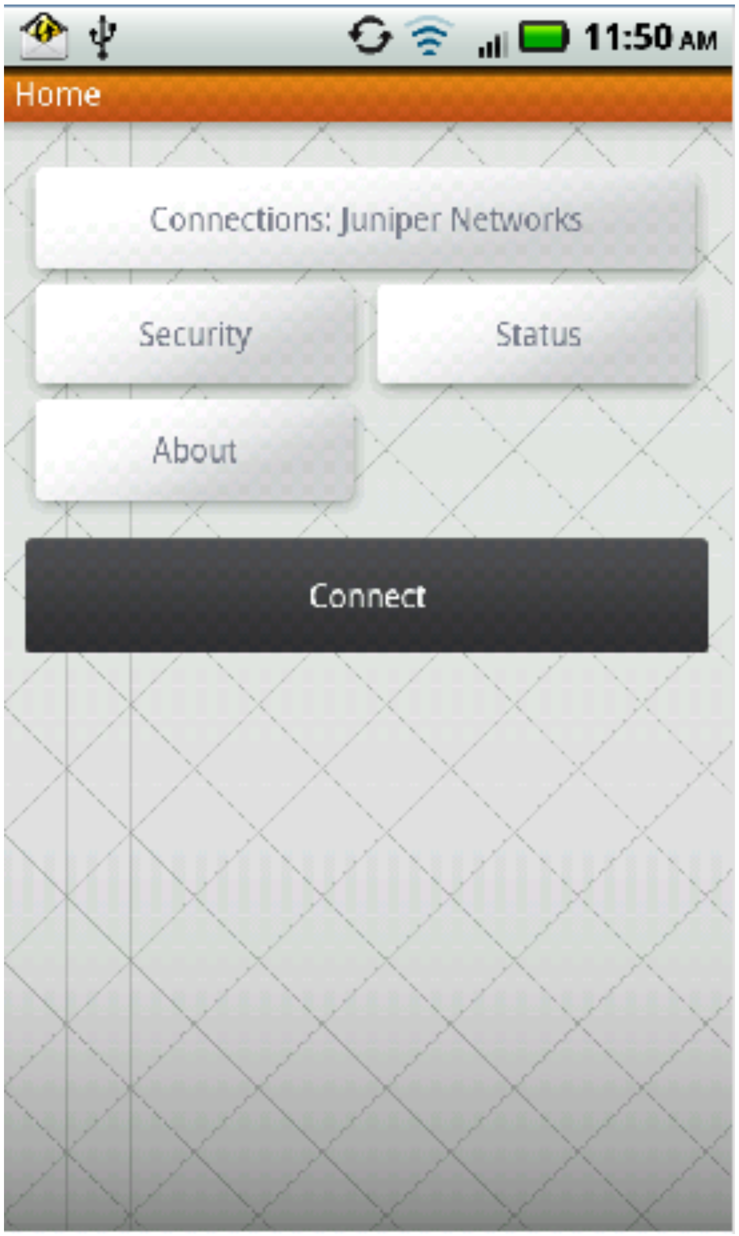

Edit Connection

Juniper Networks

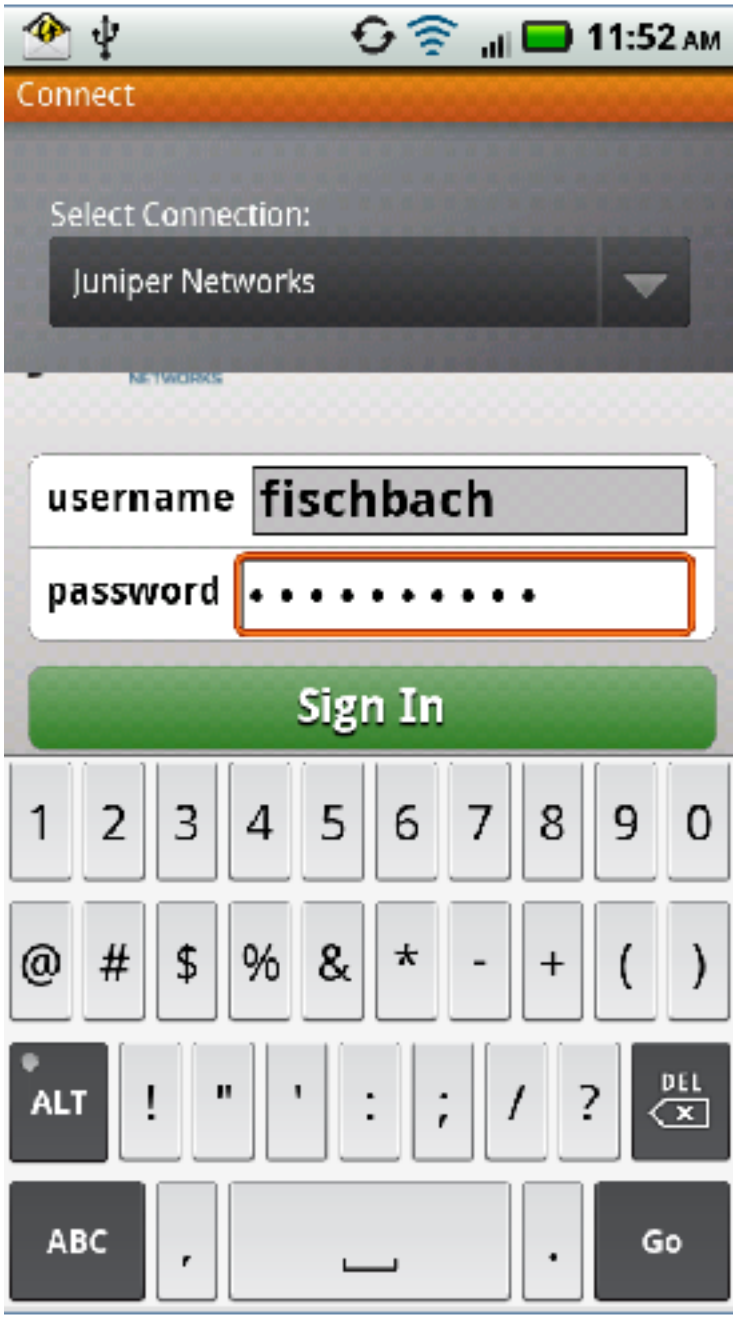
https://svl-pulse.juniper.net/mol

Uses Certificate ✓

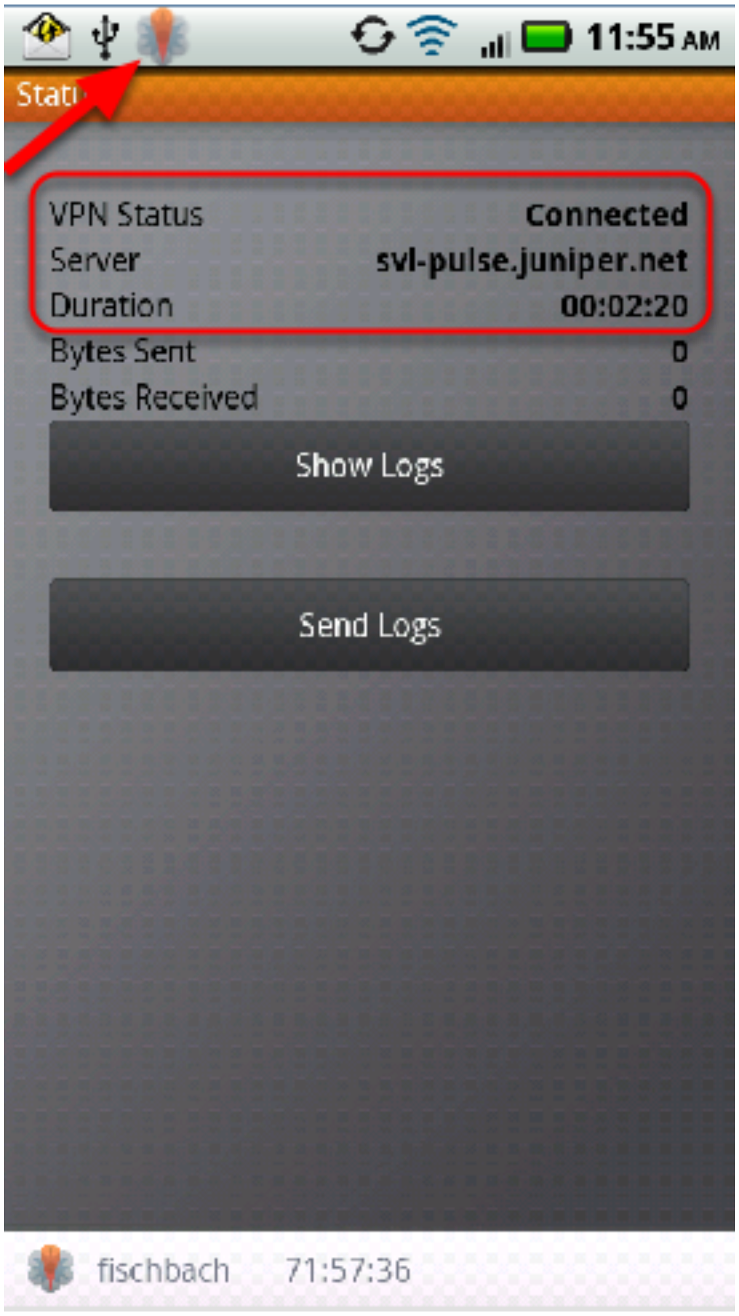
Connect to the Intranet by selecting **Connect**



Enter your login ID and SecureID pin and passcode.

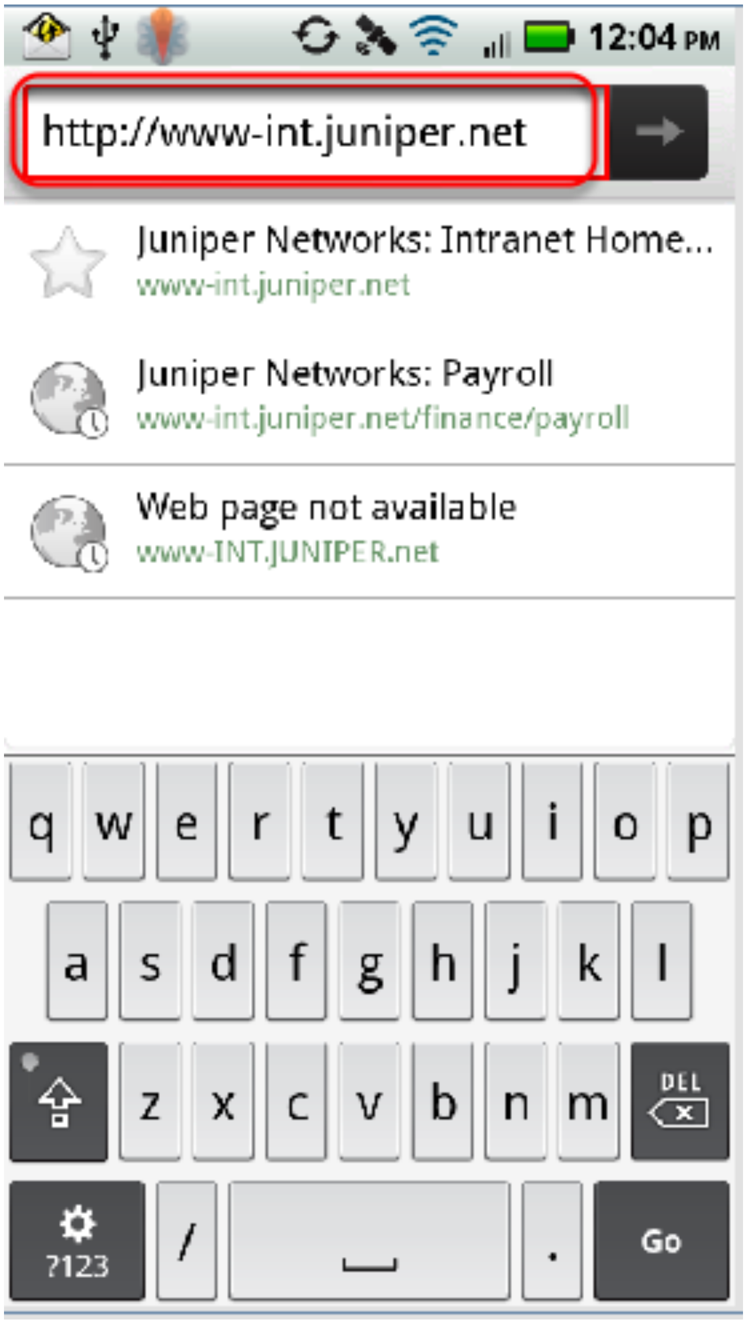


You will now be connected to the SA/MAG and a full Layer 3 tunnel will be established. This can be verified by the Pulse icon at the top of the screen and through the Status screen. Select **Status**



| | |
|---|---|
| VPN Status | Connected |
| Server | svl-pulse.juniper.net |
| Duration | 00:02:20 |
| Bytes Sent | 0 |
| Bytes Received | 0 |

Now, test the connection to an Intranet resource.  Lets use http://www-int.juniper.net.  Open the standard browser and navigate to the Intranet Page's URL.

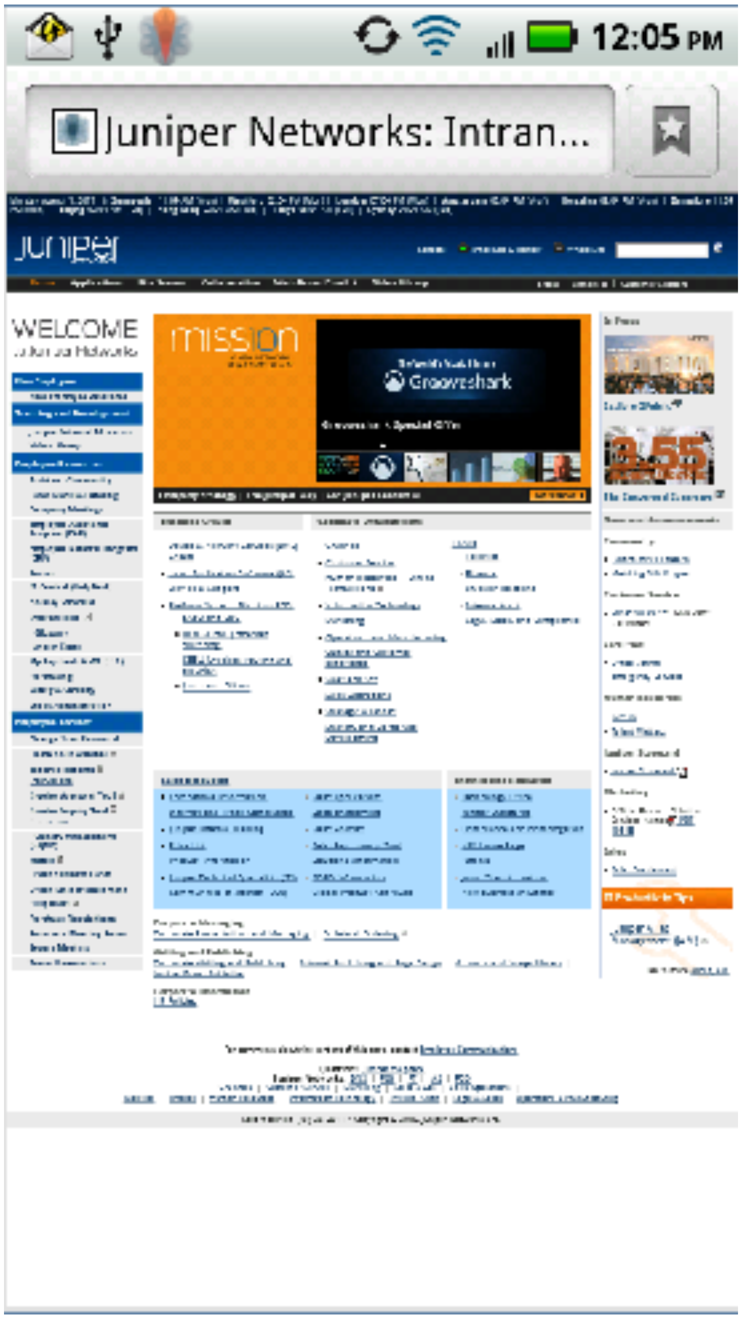

12:04 PM

http://www-int.juniper.net

Juniper Networks: Intranet Home...
www-int.juniper.net

Juniper Networks: Payroll
www-int.juniper.net/finance/payroll

Web page not available
www-INT.JUNIPER.net



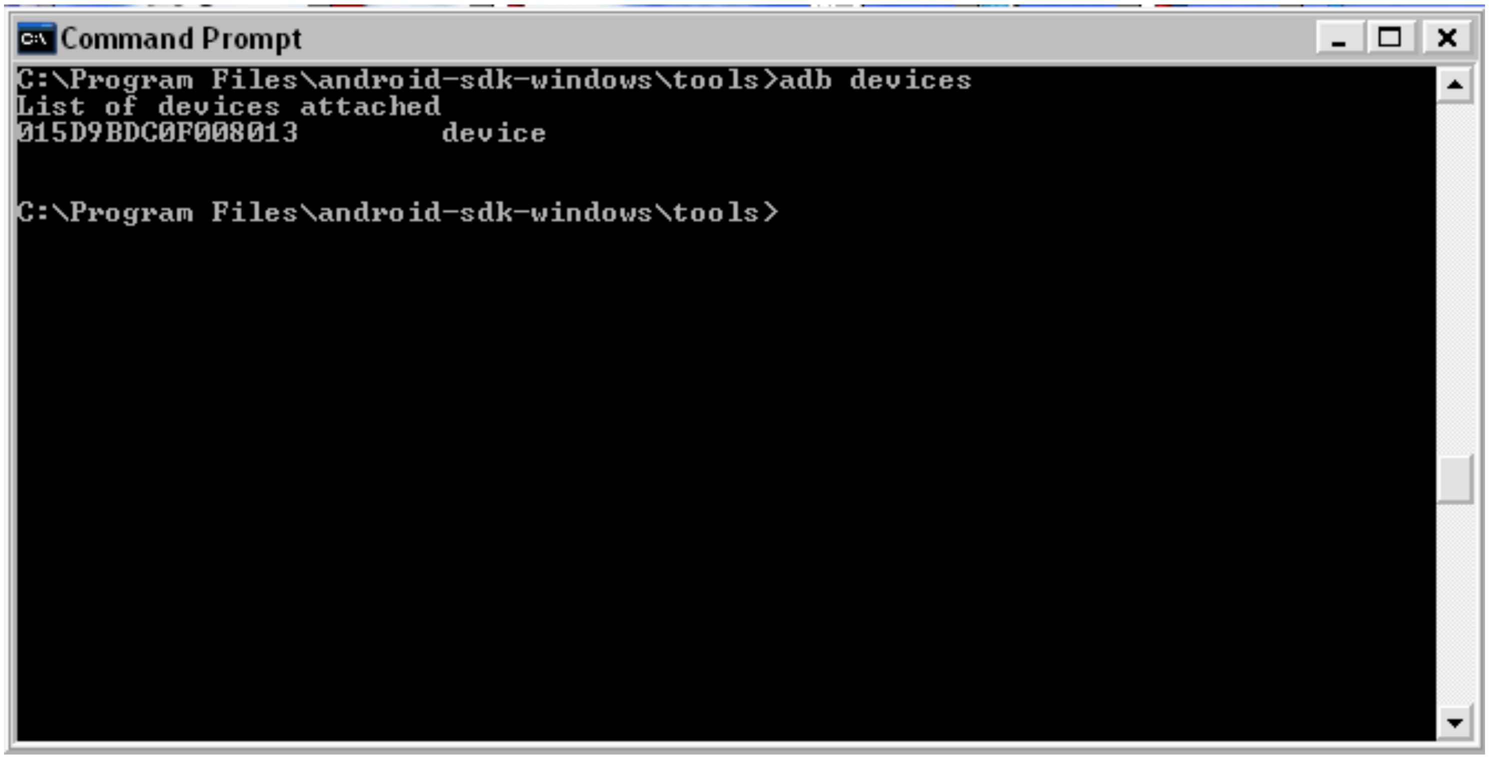
## Appendix A - Other ADB and Shell Commands

There are other useful commands that you can use to see the status of the devices. Here are a few that can be used for configuration and troubleshooting purposes.

**Querying for Devices**

**> adb devices**

Here's an example showing the **devices** command and its output:



```
C:\Program Files\android-sdk-windows\tools>adb devices
List of devices attached
015D9BDC0F008013        device


C:\Program Files\android-sdk-windows\tools>
```

**Installing an Application**

You can use adb to copy an application from your development computer and install it on adevice. To do so, use the **install** command. With the command, you must specify the path to the .apk file that you want to install:

> **adb install <path_to_apk>**

**Copying Files to or from a Device**

You can use the adb commands **pull** and **push** to copy files to and from a device. Unlike the install command, which only copies an .apk file to a specific location, the pull and push commands let you copy arbitrary directories and files to any location in a device.

To copy a file or directory (recursively) *from* the device, use:

**> adb pull <remote> <local>**

To copy a file or directory (recursively) *to* the device, use:

> **adb push <local> <remote>**

In the commands, <local> and <remote> refer to the paths to the target files/directory on your development machine (local) and on the device (remote).

Here's an example:

> **adb push foo.txt /sdcard/foo.txt**

In some cases, you might need to terminate the adb server process and then restart it. For example, if adb does not respond to a command, you can terminate the server and restart it and that may resolve the problem.

To stop the adb server, use:

> **adb kill-server**
You can then restart the server by issuing any adb command or use:
> **adb start-server**