ANACOM, INC.
1996 Lundy Ave, San Jose, CA 95131, USA
P/N 34754 Rev 04
Phone: 408.519.2062 FAX: 408.519.2063

How to run Supervisor on Linux

This document describes how to run the AnaCom, Inc. Supervisor NMS program (specifically version 10.1.11 or later,) which is a Windows program, on a platform running Linux. This document is not intended to imply an endorsement of one system over the other.

The directions provided here have been tested on a laptop running Ubuntu 14.04 64-bit Desktop version; mileage on a platform using another Linux distro may vary.
See: **http://www.ubuntu.com/download/desktop**

The particular desktop the user wishes to use should not make any difference, but our setup uses Cinnamon 2.8.6 in lieu of the standard Unity desktop that Ubuntu uses.
See: **http://www.webupd8.org/2015/06/cinnamon-26-officially-released-video.html**

Assuming the user has an Ubuntu or a compatible Linux system in front of him, and some knowledge of Linux and how packages are installed, we can proceed.

Much of this work will be done with commands entered into a terminal window, using root privileges. The length of this addendum might lead the reader to believe it is going to be difficult to install Supervisor on a Linux platform, but it's length is mostly due to the various options the user has at his disposal; the process is not difficult.

It is necessary to install Wine, which allows for Windows applications to run on Linux.
See: **https://www.winehq.org/**

Most Linux installations today are running a 64-bit version of the OS. Supervisor is a 32-bit program however, but installing Wine should bring in the necessary i386 32-bit libraries automatically.

To get the latest stable version, use the "Ubuntu Wine Team" repository:
ppa:ubuntu-wine/ppa

To install Wine:
```
sudo add-apt-repository ppa:ubuntu-wine/ppa
sudo apt-get update
sudo apt-get install wine
```

More than likely the most recently released stable version of Wine is desired. If a staging, development, or nightly version is desired however, then it may be turn out that not all the necessary libraries will be brought in automatically, and missing libs will then have to be installed manually.

Once Wine is installed, we recommend installing **PlayOnLinux**, and using it's comfortable interface to install Windows applications and running them, but it is not required.

Using PlayOnLinux provides for some added security and isolation between Windows programs running on a Linux platform by creating a virtual drive in which installed programs will exist. PlayOnLinux can create an individual virtual drive for every Windows program that is installed. See: **http://wiki.winehq.org/PlayOnLinux**, **https://www.playonlinux.com/en/**

Now for some important background information about how Supervisor works. Supervisor finds AnaCom devices on a user's Local Area Network by using tools such as the Bootp protocol (using the same UDP ports as DHCP – 67 and 68) and ping. The BootP protocol is used to send a broadcast to all AnaCom devices on port 68 requesting a response to be returned to port 67. Ping is used to check if there is an IP address collision for any of the found devices.

A normal user on a Linux platform is not permitted to open sockets numbered below 1024, nor open raw sockets, (necessary for using ping from a program.)

Fortunately, the solution is simple: grant Wine extended privileges. The user will need root access temporarily, but afterwards Supervisor will NOT be running with root privileges. We can use the **setcap** command to do this. For more information, see: http://linux.die.net/man/7/capabilities. If **setcap** is not installed:
sudo apt-get install libcap2-bin

Assuming the stable version of wine is being used, enter the following commands:
sudo setcap cap_net_raw+epi /usr/bin/wine-preloader
sudo setcap cap_net_bind_service=+ep /usr/bin/wine-preloader

Assuming a wine-staging or development version is being used, then a different set of commands will need to be entered, for example:
sudo setcap cap_net_raw+epi /opt/wine-staging/bin/wine-preloader
sudo setcap cap_net_bind_service=+ep /opt/wine-staging/bin/wine-preloader

Note: when wine is updated, these commands will have to be re-entered, but otherwise, the elevated privileges awarded to Wine by **setcap** will be persistent across a reboot.

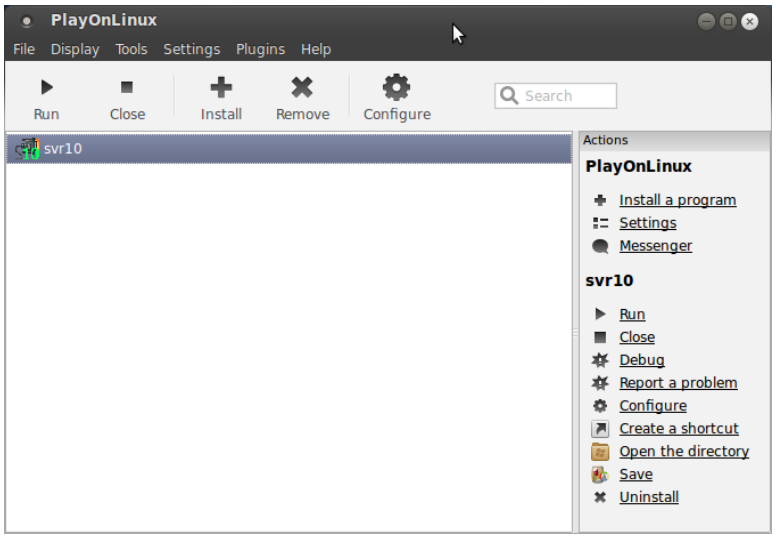
Note: Unusual modifications to the Linux iptables/firewall setup, such as by allowing your machine to serve as a WiFi hotspot, can disable the user's ability to open privileged TCP/UDP ports on a wired Ethernet port in a seemingly unrelated application. For more information, see: https://help.ubuntu.com/community/IptablesHowTo

Now, we are ready to install Supervisor! Download from the AnaCom, Inc. website.

See: http://anacominc.com/svisor10.setup.exe

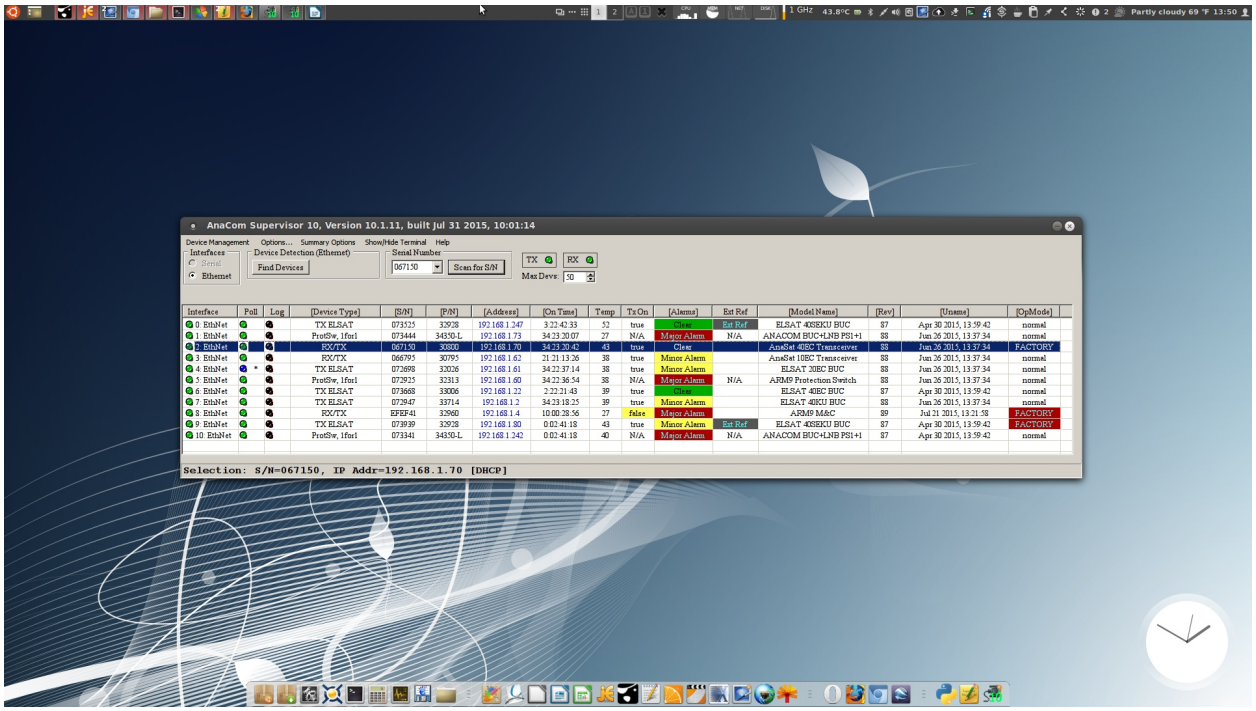
We can now use PlayOnLinux to run the install program and create a shortcut for Supervisor. When the installer is near the end and asks if you would like to Launch Supervisor, we suggest you not do this; let PlayOnLinux finish it's job.

Example of the PlayOnLinux interface after it has installed Supervisor successfully:



Note: there is an incompatibility in Supervisor, (if older than version 10.1.11,) with Linux - when finding units to monitor, error messages in Supervisor's terminal window may occur indicating that it is not finding entries in the system's ARP table for units it has found. Wine does not appear to provide complete access to the Linux ARP table for a Windows program. It is best to just ignore these messages, (printed to the Terminal display,) as they do not appear to pose a problem. We recommend updating Supervisor to 10.1.11 or later, which will eliminate the issue altogether.

Once installed we can run Svr10 from the PlayOnLinux GUI by selecting Svr10 and then clicking on Run. Below is a screen shot of our Linux laptop that is running Supervisor 10.



Once it has been installed, it is easy to run Svr10 from the command line as well, note that the option play is preceded by two dashes, and not just one:
playonlinux --run svr10

Running Supervisor (Svr10) on Linux to monitor Ethernet connected devices without elevated permissions

It is certainly possible to run Supervisor (version 10.1.11 or later) on a Linux platform without the benefit of either ping or the use of Bootp (DHCP) which is used to find units without knowledge of the IP addresses of the devices to be searched for.

First, we must disable Supervisor's use of ping as the program is not able to determine automatically when ping is not going to work. Using the terminal window, (select Show/Hide Terminal in the main window menu bar,) enter the internal command "**>ping disable**" without quotes and that should take care of it. This setting will be persistent across reboots. Ping is not critical to the use of Supervisor; it serves only to determine if the device has an IP address that is in conflict with another device on the LAN that has the same IP address. This is unlikely in a DHCP-managed LAN, but can certainly happen where many units have been assigned a static address.

With ping disabled, device discovery using IP addresses runs faster.

Secondly, if Bootp is not available because Supervisor could not open UDP port 67 for listening, then the normal method of finding units is disabled, and we will have to find units by scanning for their IP addresses instead.

On a Linux machine, we can quickly get a list of the IP addresses of all AnaCom devices present on the LAN using the Linux tool **netdiscover**. Using the script below, we can get such a list ready to copy & paste into Supervisor:

sudo netdiscover -i eth0 -r 192.168.1.0/24 -P | grep -i "0c:ef:7c" | cut -d' ' -f2

Note: if netdiscover is not present, it may have to be installed:

sudo apt-get install netdiscover

It might be necessary to specify a different interface or a different route depending on the user's platform running Supervisor, and his LAN IP address range. Note that the Ethernet MAC prefix for AnaCom, Inc. device is 0c:ef:7c.

Netdiscover is an interesting tool. It can discover units actively by sending ARP broadcasts. See: http://manpages.ubuntu.com/manpages/maverick/man8/netdiscover.8.html

Once this list has been obtained, (netdiscover runs very quickly,) we can paste it into the "S/N List" window in Supervisor. Beginning with version 10.1.11, this feature will accept IP addresses for devices as well as serial numbers.

Under the menu titled "Device Management" in the main window, select "Find from S/N List." Paste the list of IP addresses into the open edit window. Then, select "Find Listed Devices." Supervisor will now look for AnaCom-built devices with those serial numbers. It should find them and begin monitoring them very quickly.

Note: without elevated permissions that allow ping and Bootp to work, Supervisor will be unable to recover a device that has an IP address that is colliding with another unit using the "**>snreset**" command. This problem should not occur on a DHCP-managed LAN, but on a LAN with devices that have been assigned static IP addresses, it could.

Forwarding the Bootp ports on Linux

Beginning with Svr10 version 10.1.11, if Supervisor is not able to use port 67 for listening and port 68 for broadcasting Bootp discovery packets, the user can choose instead to use an alternate set of ports, 5067 and 5068. This is done using a new internal command >bootp_alt.

Example:

>bootp_alt enable	- enables use of the alternate UDP ports for device scanning
>bootp_alt disable	- resets the Svr10 device scanning function to ports 67 and 68
>bootp_alt	- gives the current Bootp_alt status

This choice is persistent across a restart of Svr10. Upon enabling, (or disabling,) the alternate ports however, it will be necessary to restart Svr10 before the change takes effect. When Supervisor (version > 1.10.11) starts up when the alternate Bootp ports are being used, a message to this effect will be printed in the Terminal window.

Example:

Bootp binding to alternate ports: listen=5067, client=5068

Now, it will be necessary to forward incoming UDP packets on port 67 to port 5067, and forward outgoing UDP packets on port 5068 back to port 68. This is done using the program iptables that serves to configure the Linux kernel firewall. See the four commands below:

```
sudo iptables -t nat -A PREROUTING -i eth0 -p udp --dport 67 -j REDIRECT --to-port 5067
sudo iptables -I INPUT -p tcp --dport 67 -j ACCEPT
sudo iptables -I INPUT -p tcp --dport 5067 -j ACCEPT
sudo iptables -t nat -A OUTPUT -p udp --dport 5068 -j DNAT --to-destination :68
```

Note: these iptables rules will not be persistent across a reboot. For convenience sake, we suggest creating an executable shell script named iptables-svr10, that looks like this:

```
#!/bin/bash
iface=${1:-eth0}
sudo iptables -t nat -A PREROUTING -i $iface -p udp --dport 67 -j REDIRECT --to-port 5067
sudo iptables -I INPUT -p tcp --dport 67 -j ACCEPT
sudo iptables -I INPUT -p tcp --dport 5067 -j ACCEPT
sudo iptables -t nat -A OUTPUT -p udp --dport 5068 -j DNAT --to-destination :68
```

This will permit the script to run without being called from a terminal window, and will graphically prompt the user for the root password. To make a script executable, example:
chmod u+x iptables-svr10

Further, we could create another executable script called run-svr10 that looks like this, where user should be replaced with an actual user name:

```
#!/bin/bash
sudo /home/user/bin/iptables-svr10 ${1:-eth0}
playonlinux --run svr10
```

Example usage:

```
$ run-svr10 eth1
```

Note: in the first iptables rule given above, we assumed that the name of the Ethernet port was eth0, but this might not necessarily be the case. For example, it could be something like enp2s0. Use the **ifconfig** command to dump all active Ethernet interfaces, before choosing the correct interface that is connected to the desired network.

Using **sudo** normally requires the root password to be given, but we can add this requirement and allow a specific user to execute the script run-svr10 from the command line by adding permanent sudo permissions using the program **visudo**:

```
sudo visudo
```

After the line/entry for the sudo group, we can add a line that looks like:

```
user    ALL=(ALL) NOPASSWD: /home/user/bin/iptables-svr10
```

replacing “user” with the desired user’s name, and the proper path to iptables-svr10 will have to be used as well. For more information, see: https://help.ubuntu.com/community/Sudoers.

Playing around with iptables could potentially cause a problem because you will be playing with your system's firewall at a low level. Potential problems can be mitigated however. Before adding rules to the firewall using iptables, it can easily be backed up:
sudo iptables-save > iptables.backup

To restore this snapshot:
sudo iptables-restore < iptables.backup

The iptables nat rules can be flushed using the command:
sudo iptables -F -t nat

To view the currently active rules that iptables is using at any time:
sudo iptables -t nat -L -v -n

Example output from the above command:

```
$ sudo iptables -t nat  -L -v -n
Chain PREROUTING (policy ACCEPT 5 packets, 397 bytes)
 pkts bytes target     prot opt in     out     source               destination
   0     0 REDIRECT   udp  --  eth0   *      0.0.0.0/0            0.0.0.0/0           udp dpt:67 redir ports 5067

Chain INPUT (policy ACCEPT 5 packets, 397 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 2 packets, 168 bytes)
 pkts bytes target     prot opt in     out     source               destination
   0     0 DNAT       udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:5068 to::68

Chain POSTROUTING (policy ACCEPT 2 packets, 168 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Since these changes will not be persistent, a reboot will also clear out whatever mess might have been created. For more information on iptables and the Linux kernel firewall, see:
https://en.wikipedia.org/wiki/Iptables

Note: it is only necessary to make changes to the Linux firewall to permit the use of bootP to scan for devices to monitor; it is not necessary to maintain these changes to monitor the devices when Supervisor is doing periodic polling. Polling is done using ports 5001, (for sending commands,) and port 5000, (for receiving responses from polled devices.) Changes made to the firewall using a script such as iptables-svr10 described above will normally remain active after Supervisor has been shut down, until they are explicitly deleted, or flushed.

In summary, what might be best from a security standpoint is to use iptables to modify the firewall for Svr10 and use setcap to allow Wine access to raw ports so that ping will work, or simply disable the use of ping with the internal command **>ping disable** entered in Svr10's Terminal window command field.

The question has probably occurred to the reader – why don't we just run Supervisor as root e.g., sudo wine svr10.exe, and be done with it? There are two reasons for not doing this:

1. Running Wine as root is strongly discouraged from a security perspective. It permits a Windows application root access to the Linux platform.

2. For reasons that are not clear, Supervisor does not run reliably under root. Svr10 is heavily multi-threaded, and when running as a root process, not all the threads start successfully.

Monitoring devices in multiple subnets

It is possible to monitor devices that are in multiple subnets on a LAN, for example: 192.168.1.0//24 and 169.2254.0.0/16.

We use **ifconfig** to create an alias on eth0 for accessing a second subnet, before starting Supervisor. In the following example, we create the alias eth0:0, and assign IP address 169.254.78.205 for our new interface. We get a /16 netmask; I think this happens because ifconfig recognizes that this address falls into the APIPA address space which is a class B subnet.

```
$ sudo ifconfig eth0:0 169.254.78.205 up
```

The network configuration that results will be similar to this:

```
$ ifconfig
eth0      Link encap:Ethernet  HWaddr 80:fa:5b:02:cb:7a
          inet addr:192.168.1.205  Bcast:192.168.255.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:20 Memory:f7e00000-f7e20000

eth0:0    Link encap:Ethernet  HWaddr 80:fa:5b:02:cb:7a
          inet addr:169.254.78.205  Bcast:169.254.255.255  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          Interrupt:20 Memory:f7e00000-f7e20000
```

Note: This will allow Supervisor to monitor and control units in these two subnets, however, the program is limited to doing this successfully only if the we clear all known devices and re-acquire devices using **Find Devices**. I think this might be due to that we do not store the name of the Ethernet interface used for each device in the program's .ini file.

Running Supervisor (Svr10) to monitor Serial-port connected devices

This can be done but at the present time, it appears that Svr10 running in a Linux environment cannot properly detect the presence of any serial ports. Prior to Svr10 Version 10.1.30, it was not possible to use a serial port that the user knows is present, but that Supervisor could not find on it's own – the Serial Interface radio button would remain grayed out.

Beginning with Version 10.1.30, there is an internal command that can be entered in Supervisor's ASCII terminal window command field, to add any serial port desired ranging from COM1 to COM256. Example:

```
>serial add com1
```

Note that this is a command read by Supervisor, not by any connected ODU, the > symbol prefixing the command determines that.

Preliminaries and Creating a readable serial COM port in Linux

First, we need to add the user that will be using Supervisor to the dialout group, example:

```
$ sudo adduser <user_name> dialout
```

Before we launch Supervisor however, we need to create the desired COM port.

Linux accesses devices as if they were files in the root directory under /dev. For example, serial ports that came installed with the system will be called /dev/ttyS0, /dev/ttyS1, /dev/ttyS2, etc. If we plug in a USB serial port adapter, we will see a new file appear, such as /dev/ttyUSB0.

We need to create a symbolic link for this device file where Supervisor has been installed, in the dosdevices directory, example:

```
$ cd  /home/ron/.PlayOnLinux/wineprefix/svisor10/dosdevices
$ ln  -s  /dev/ttyUSB0  com1
```

Note: when the USB serial adapter is pulled, the USB dev file will disappear, but the symbolic link in dosdevices will remain.

On some systems, the permissions might not be adequate, so, using a text editor, we should add udev rule, example:

```
$ sudo nano /etc/udev/rules.d/ttyUSB.rules
```

We then add a line to this file that looks like:

```
KERNEL=="ttyUSB0" SYMLINK+="%k" GROUP="ron" MODE="0666"
```

Now, we are ready to plug in our USB serial adapter, and start Supervisor!

Once running, we can open the terminal window, and in the command field enter the command:

```
>serial add com1
```

Now, the Serial Interface radio button can be selected, and the serial port COM1 will appear. We can now search for devices that are connected via the serial port adapter.

Note: online references for using serial ports in a Windows application running under Wine:

1. http://g8ogj.org/files/Using%20USB%20serial%20ports%20under%20wine%20howwto%20ipb.pdf

2. https://onetransistor.blogspot.com/2015/12/wine-serial-port-linux.html