|  |  |
| --- | --- |
| PROJECT: | FusionOS Start Up Process Notes |
| SUBJECT: | How the Recovery Jumper Works to reset the SSID |
| UPDATED: | 3/28/2019 2:54:00 PM |

A Note to the Reader:

Forgive me as I walk through this process. I am documenting it here for myself to follow as well as you. By next week, I will have forgotten how it works until the next time a problem crops up.

This document is based on the v1.2.0a RC3 Build-12 version of the FusionOS software. If changes are made that make the information contained herein incorrect, please make sure that this document gets updated or superseded.

It All Starts with a File...

There is a file in the /usr/Fusion/etc directory called ssid\_set.py. This file is invoked in two known places:

1. ***rc.local*** 
   1. Distributed in /usr/Fusion/etc, it is copied to the system /etc directory during installation
   2. rc.local is run at the end of system boot-up. It starts-up various services that the Fusion uses (such as hostapd and dnsmasq), and calls ssid\_set.py before launching the FusionOS runtime
2. ***admin.js*** 
   1. Located in /usr/Fusion/FusionServer/Build/app/routes
   2. admin.js is a component of the FusionOS GUI and handles user requests. It calls ssid\_set.py if the user wants to change the SSID and/or PassKey of the Fusion’s Access Point.

ssid\_set.py is primarily used to set the SSID and PassKey of the Fusion’s on-board WiFi asset when it is operating in access point mode. It recognizes four command line arguments including one to reset the SSID and PassKey to the defaults. Currently, ssid\_set.py is also responsible for looking for the presence of the System Recovery Jumper and will force the Access Point credentials back to the defaults.

The System Recovery Jumper:

The Fusion has eight (8) Digital I/O Ports and the System Recovery Jumper is a wire used to connect the data line of Port 0 to the data line of Port 7. The Digital Port inputs are floating and it cannot be guaranteed if the input is high or low, so we use one port as an output to SET a level and look for that state on the other port which is configured for input. If we can waggle that signal between high and low 10 times and read the result with no errors, we accept that the jumper is present. If the jumper is present, a status flag is set so that ssid\_set.py will force the default Access Point credentials.

While ssid\_set.py is running, it is unable to interact with the user via the GUI so it is not able to present on-screen prompts to tell the use that the jumper was detected, to remove the jumper, or even if there was a problem with the credentials being requested. It uses the two programmable LEDs (Blue and Yellow) on the Fusion as indicators, but it does not have any return codes back to the calling JavaScript code to prompt or message the user.

If the jumper is found, ssid\_set.py will loop [forever] waiting for the jumper to be removed.

Recovery Jumper Detection Code:

The following code is taken from the ssid\_set.py program:

#---------------------------------------------------------------------------

# 1) Configure the two digital ports used by the recovery jumper

#

f = Fusion.driver()

f.digitalState(f.D0, f.INPUT)

f.digitalState(f.D7, f.OUTPUT)

# 2a) Assume that the jumper is calling for a recovery reset and check to

# see if the jumper is present 10 times. If any time shows the jumper

# is NOT present, then we aren't in recovery mode.

#

external\_reset = True

for i in range(10):

f.digitalWrite(f.D7, 1)

time.sleep(0.001)

if (f.digitalRead(f.D0) != 1):

external\_reset = False

break;

time.sleep(0.001)

f.digitalWrite(f.D7, 0)

if (f.digitalRead(f.D0) != 0):

external\_reset = False

break;

# 2b) Finished checking; if external\_reset requested, light the two LEDs and

# wait for the jumper to be removed before proceeding

#

time.sleep(0.1)

if (external\_reset == True):

f.setLED(0, 1)

f.setLED(1, 1)

f.digitalWrite(f.D7, 1)

while (f.digitalRead(f.D0)): pass

System Start-Up:

As its final action in starting up, Linux passes control to the /etc/rc.local file. Everything involved with configuring the system to support the Fusion’s operation. This includes configuring the routing tables to pass traffic between the on-board wired Ethernet and WiFi assets, switching the internal WiFI asset as an Access Point, configuring the DNS and DHCP clients, setting up the dnsmasq, setting up the WiFI SSID and PassKey, and making sure that the necessary services are running. It then launches the FusionServer.

It is worth including rc.local in this document since between rc.local and ssid\_set.py, most of the system configuration and reconfiguration decisions are made and processed. This pair of files is also a key pre-existing link between the FusionOS and the processing system.

rc.local:

#!/bin/bash

#

#===============================================================================

# File: rc.local (Fusion Version)

#

# This script is executed at the end of each multiuser runlevel.

# Make sure that the script will "exit 0" on success or any other

# value on error.

#

# In order to enable or disable this script just change the execution

# bits.

#

#-------------------------------------------------------------------------------

# REVISION HISTORY:

# 03-Apr-18 <jwa> - Added host name to dns service and TCP Port redirect

#

#===============================================================================

sudo invoke-rc.d mongodb start

sudo invoke-rc.d hostapd start

sudo invoke-rc.d dnsmasq start

sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip\_forward"

sudo iptables -t nat -A POSTROUTING -o wlan1 -j MASQUERADE

sudo iptables -A FORWARD -i wlan1 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT

sudo iptables -A FORWARD -i wlan0 -o wlan1 -j ACCEPT

sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT

sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT

sudo iptables -t nat -I PREROUTING --src 0/0 --dst 192.168.50.1 -p tcp --dport 80 -j REDIRECT --to-port 8080

sudo python /usr/Fusion/etc/ssid\_set.py -b

sudo dhcpcd

# Place the location of the node server app below

cd /usr/Fusion/FusionServer/Build

#sudo forever start server.js

sudo NODE\_ENV=production forever start server.js

cd /usr/Fusion/lib/noVNC

sudo ./utils/launch.sh &

# Print the IP address

\_IP=$(hostname -I) || true

if [ "$\_IP" ]; then

printf "My IP address is %s\n" "$\_IP"

fi

exit 0