JVN Tool Suite

User Installation Guide

March 8, 2024

JVN Tool Suite v13.4.19 Volume 1

JVN Communications Inc.
2511 Fire Rd., Suite A-4
Egg Harbor Township, NJ 08234
609-569-9255
www.jvncomm.com

TABLE OF CONTENTS

1 Introduction

The JVN Tool Suite provides an integrated capability for testing a wide range of Air Traffic Control (ATC) systems that manage the National Air Space (NAS). These tools are used in various laboratories of the Federal Aviation Administration (FAA) William J. Hughes Technical Center (WJHTC) during all phases of testing, as well as, research and sustainment efforts. Capabilities of the tool suite include scenario development, simulation of the operational environment, replay of live recorded data, system command injection, and data reduction and analysis. The tools can be configured to operate in combinations of real and virtual system interfaces for EnRoute and Terminal environments, including En Route Automation Modernization (ERAM), Standard Terminal Automation Replacement System (STARS), Data Communications (DataComm), Time Based Flow Management (TBFM), Terminal Flight Data Management (TFDM), and Traffic Flow Management System (TFMS).

2 Software Installation

This document details the process of installing the JVN Tool Suite including Airspace Visualization Display (AViD), Dynamic Simulation (DYSIM), Graphic Simulation Generation Tool (GSGT), Simulation Driver Radar Recorder (SDRR), SiteShadow, and Lab Utilities products, as well as prerequisite software packages on a linux platform. The target systems should be processors with Debian version 11, Slackware version 14.0 (32 bit), or Slackware version 14.2 (64 bit) operating system (OS) installed. These processors should be configured with network access and all necessary drivers.

2.1 Qt Graphics Library

The Qt Graphics Library must be installed on each processor that will be running any component of the JVN Tool Suite.

a. Log onto each processor as the root user.
b. Copy the Qt Install Package into the /tmp directory.
c. From the /tmp directory, extract the Qt Install Package.

For Slackware 14.0, enter:

```
> installpkg ./jvn-qt-5.7.1-slack14-runtime.tgz
```

NOTE: *The final installation location for the revision should be /usr/lib/qt-5.7.1/.*

For Slackware 14.2, enter:

```
> installpkg ./jvn-qt-5.9.0-slack14.2-x64-runtime.tgz
```

NOTE: *The final installation location for the revision should be /usr/lib64/qt-5.9.0/.*

NOTE: *For Debian processors, the Qt installation is included in the JVN Software Dependencies package, see section 2.4. The final installation location for the revision should be /usr/lib64/qt-6.4.1/.*

2.2 Java Runtime Environment

The Java Runtime Environment (JRE) must be installed on each processor that will be running any component of the JVN Tool Suite.

a. Log onto each processor as the root user.
b. Copy the JRE Install Package into the /tmp directory.
d. From the /tmp directory, uncompress and extract the JRE Install Package.

For Slackware 14.0, enter:

```
> gunzip jre-8u191-linux-i586.tar.gz
> tar –xvf jre-8u191-linux-i586.tar –C /usr/lib
```

For Slackware 14.2, enter:

```
> gunzip jre-8u191-linux-x64.tar.gz
> tar –xvf jre-8u191-linux-x64.tar –C /usr/lib64
```

NOTE: *For Debian processors, JRE is handled via the package manager and dependencies are specified in the JVN Tool Suite software package.*

2.3 Oracle Virtual Box

Oracle Virtual Box may be installed on the processor that will be running SDRR, if desired. It may be beneficial to have virtual machines configured as En Route Data Distribution Systems (EDDS) or Terminal Data Link Service (TDLS).

a. Log onto each processor where SDRR will run.
b. As the root user, copy the VirtualBox Install Package into the /tmp directory.

c. From the /tmp directory, extract the Install Package file, enter:

For Debian, enter:

```
> apt install ./virtualbox-7.0_7.0.6-155176_Debian_bullseye_amd64.deb
```

For Slackware 14.0, enter:

```
> ./VirtualBox-5.0.12-104815-Linux_x86.run install
```

For Slackware 14.2, enter:

```
> installpkg ./virtualbox-5.0.40.tar.gz
```

2.4 JVN Software Package Dependencies

The JVN software package dependencies and tool libraries must be installed on each processor that will be running any component of the JVN Tool Suite.

a. Log onto each processor as the root user.
b. Copy the JVN Libraries Install Package into the /tmp directory.
c. From the /tmp directory, extract the Install Package files.

For Debian 11, enter:

```
> apt install ./jvn-runtime_2.0.0_amd64.deb
```

For Slackware 14.0, enter:

```
> installpkg ./libjvn-Slack14.0.tgz
```

For Slackware 14.2, enter:

```
> installpkg ./libjvn-Slack14.2.tgz
```

2.5 JVN Tool Suite Package

The JVN Tool Suite must be installed on each processor that will be running any component of the JVN Tool Suite.

a. Log onto each processor as the root user.
b. Copy the JVN Tools Install Package into the /tmp directory.
c. From the /tmp directory, extract the Install Package files.

For Debian 11, enter:

```
> apt install./jvn-13.4.20-debian.deb
```

For Slackware 14.0, enter:

```
> installpkg ./jvn-slack14.0-13.4.20.tgz
```

For Slackware 14.2, enter:

```
> installpkg ./jvn-slack14.2-13.4.20.tgz
```

2.6 License Files

Each processor that will be running one or more components of the JVN Tool Suite, must have license files installed for those components.

a. Log onto each processor as the root user.
b. Copy the appropriate license file(s) into the /etc directory.

3 Configuration

3.1 Create Subinterfaces

Network subinterfaces may be configured on the processor that will be running SDRR, if desired. It may be beneficial to have addition interfaces for communication with ERAM in a Box (EIB) processors or locally installed virtual machines.

3.1.1 Slackware Processors

To create subinterfaces on processors installed with a Slackware OS:

a. Log onto each processor where SDRR will run.
b. As the root user, define the subinterface addresses, enter:

```
> cd /etc/rc.d
> vi rc.subinterfaces
```

c. Add a line for each subinterface in the following format:

```
  ifconfig  ethx:y aaa.bbb.ccc.ddd   netmask   255.255.255.0
```

where x is the number of the Ethernet device, y is the number of the subinterface and aaa.bbb.ccc.ddd is the IP address of the new subinterface.

d. Once all subinterfaces have been defined, enter:

```
  :wq!
```

e. Ensure the rc.subinterfaces file is executable. If necessary, enter:

```
> chmod +x rc.subinterfaces
```

f. At the command line, enter:

```
> vi rc.local
```

g. Add the following lines:

```
  if [ -x /etc/rc.d/rc.subinterfaces ]; then
     . /etc/rc.d/rc.subinterfaces
  fi
```

h. Once finished, enter:

```
  :wq!
```

i. At the command line, enter:

```
> ./rc.local
```

j. Verify that all subinterfaces were created; enter:

```
> ifconfig -a
```

k. Add the subinterface addresses into the hosts file. Enter:

```
> vi /etc/hosts
```

l. Enter the addresses defined in step c, then enter:

```
> :wq!
```

3.1.2 Debian Processors

To create subinterfaces on processors installed with a Debian OS:

a. Log onto each processor where SDRR will run.
b. As the root user, define the subinterface addresses, enter:

```
> cd /etc/network/interfaces.d/
> vi $interfaceName (ex. eno1)
```

c. Define each subinterface in the following format:

```
auto $interfaceName:y
allow-hotplug $interfaceName:y
iface $interfaceName:y inet static
       address aaa.bbb.ccc.ddd/24
```

Where $interfaceName is the name of the Ethernet device, y is the number of the subinterface and aaa.bbb.ccc.ddd is the IP address of the new subinterface.

d. Once all subinterfaces have been defined, enter:

```
  :wq!
```

e. At the command line, enter:

```
> ifdown $interfaceName && ifup $interfaceName
```

f. Verify that all subinterfaces were created; enter:

```
> ip a
```

g. Add the subinterface addresses into the hosts file. Enter:

```
> vi /etc/hosts
```

h. Enter the addresses defined in step c, then enter:

```
> :wq!
```

4 Radar Interfaces

4.1 Create a sccrdr.conf file

Create a sccrdr.conf file in /usr/local/etc based on the amount of radar cards in the serial processor (SP). The sccrdr.conf should allow for individual channels per each radar card. Below is an example of an sccrdr.conf file. Device 0 is divided into single and three channel radars. An example of a three channel radar definition is displayed under the heading DEV-0's. Text can be added following the '##' after the radar channel definitions. This text will be displayed when using the **monitor** command.

```
#
# sccrdr 4.x: all radar names must start with “srr”, “lrr”, or “asr11” no extra lines or
spaces
#
#####STANDARD##########
srr0        0/0 0/1 0/2 0/3 ##NUN1_MOD1
asr11-0     0/0
lrr1        1/0 1/1 1/2     ##NUN1_MOD2
lrr2        1/3 2/0 2/1     ##NUN1_MOD3
srr3        2/2 2/3 3/0 3/1 ##NUN1_MOD4
asr11-3     2/2
srr4        3/2 3/3 4/0 4/1 ##NUN1_MOD5
asr11-4     3/2
srr5        4/2 4/3 5/0 5/1 ##NUN2_MOD1
asr11-5     4/2
lrr6        5/2 5/3 6/0     ##NUN2_MOD2
lrr7        6/1 6/2 6/3     ##NUN2_MOD3
srr8        7/0 7/1 7/2 7/3 ##NUN2_MOD4
asr11-8     7/0
#####DEV-0’s##########
srr0:1:2:3  0/0 0/1 0/2
asr11-0:1   0/0
#####DEV-1’s##########
lrr1:1      1/0
lrr1:1:2    1/0 1/1
lrr1:2      1/1
lrr1:3      1/2
```

4.2 Create a devices file

Create a devices file in /usr/local/etc on the application processor (AP) based on the radar interfaces defined in the sccrdr.conf on the serial processors. Configuration files based on adaptations will directly reference the devices file allowing radars to be directed to the correct NUNIOs. A devices file template is included below.

```
##Format examples:
##asr4/asr9 /dev/srr0
##asr11 /dev/asr11-0:1
##lrr /dev/lrr0
##1chan lrr /dev/lrr0:1

NUN1_MOD1 /dev/null
NUN1_MOD1_CHAN1 /dev/null ##format /dev/asr11-#:1
NUN1_MOD1_CHAN2 /dev/null ##format /dev/asr11-#:2
NUN1_MOD2 /dev/null
NUN1_MOD2_CHAN1 /dev/null ##format /dev/lrr#:1
NUN1_MOD2_CHAN2 /dev/null ##format /dev/lrr#:2
NUN1_MOD3 /dev/null
NUN1_MOD3_CHAN1 /dev/null
NUN1_MOD3_CHAN2 /dev/null
NUN1_MOD4 /dev/null
NUN1_MOD4_CHAN1 /dev/null ##format /dev/asr11-#:1
NUN1_MOD4_CHAN2 /dev/null ##format /dev/asr11-#:2
NUN1_MOD5 /dev/null
NUN1_MOD5_CHAN1 /dev/null ##format /dev/asr11-#:1
NUN1_MOD5_CHAN2 /dev/null ##format /dev/asr11-#:2
NUN2_MOD1 /dev/null
NUN2_MOD1_CHAN1 /dev/null ##format /dev/asr11-#:1
NUN2_MOD1_CHAN2 /dev/null ##format /dev/asr11-#:2
NUN2_MOD2 /dev/null
NUN2_MOD2_CHAN1 /dev/null ##format /dev/lrr#:1
NUN2_MOD2_CHAN2 /dev/null ##format /dev/lrr#:2
NUN2_MOD3 /dev/null
NUN2_MOD3_CHAN1 /dev/null ##format /dev/lrr#:1
NUN2_MOD3_CHAN2 /dev/null ##format /dev/lrr#:2
NUN2_MOD4 /dev/null
NUN2_MOD4_CHAN1 /dev/null ##format /dev/asr11-#:1
NUN2_MOD4_CHAN2 /dev/null ##format /dev/asr11-#:2
NUN2_MOD5 /dev/null
NUN2_MOD5_CHAN1 /dev/null ##format /dev/asr11-#:1
NUN2_MOD5_CHAN2 /dev/null ##format /dev/asr11-#:2
```

```
NUN2_MOD6 /dev/null
NUN2_MOD6_CHAN1 /dev/null ##format /dev/asr11-#:1
NUN2_MOD6_CHAN2 /dev/null ##format /dev/asr11-#:2
NUN3_MOD1 /dev/null
NUN3_MOD1_CHAN1 /dev/null ##format /dev/asr11-#:1
NUN3_MOD1_CHAN2 /dev/null ##format /dev/asr11-#:2
NUN3_MOD2 /dev/null
NUN3_MOD2_CHAN1 /dev/null ##format /dev/asr11-#:1
NUN3_MOD2_CHAN2 /dev/null ##format /dev/asr11-#:2
NUN3_MOD3 /dev/null
NUN3_MOD3_CHAN1 /dev/null ##format /dev/asr11-#:1
NUN3_MOD3_CHAN2 /dev/null ##format /dev/asr11-#:2
NUN3_MOD4 /dev/null
NUN3_MOD4_CHAN1 /dev/null ##format /dev/asr11-#:1
NUN3_MOD4_CHAN2 /dev/null ##format /dev/asr11-#:2
NUN3_MOD5 /dev/null
NUN3_MOD5_CHAN1 /dev/null #format /dev/asr11-#:1
NUN3_MOD5_CHAN2 /dev/null #format /dev/asr11-#:2
NUN3_MOD6 /dev/null
NUN3_MOD6_CHAN1 /dev/null #format /dev/asr11-#:1
NUN3_MOD6_CHAN2 /dev/null #format /dev/asr11-#:2
DRF-1 /dev/null
DRF-2 /dev/null

NUN1_MOD6 file:///dev/null?txclock=2400&amp;rxclock=2400

ADSB_INTERFACES eth0
##This can be left out/blank since not all machines output remotely
##Format is just [remotehost]@
ADSB_OUTPUT_DEVICE

WAM_INTERFACES eth0
##This can be left out/blank since not all machines output remotely
##This line is also not added by default as it's not as common as ADSB
##Format is just [remotehost]@
WAM_OUTPUT_DEVICE

DASR_INTERFACES eth0
##This can be left out/blank since not all machines output remotely
##This line is also not added by default as it's not as common as ADSB
##Format is just [remotehost]@
DASR_OUTPUT_DEVICE
```

```
#This is mlat, not sure if this can be remote
ASDEX_MAIN_INTERFACE eth0
ASDEX_ALTERNATE_INTERFACE eth0

DASI_DEVICE1 dasi:/dev/null
DASI_DEVICE2 dasi:/dev/null
DASI_DEVICE3 dasi:/dev/null
DASI_DEVICE4 dasi:/dev/null
DASI_DEVICE5 dasi:/dev/null
DASI_DEVICE6 dasi:/dev/null
DASI_DEVICE7 dasi:/dev/null
DASI_DEVICE8 dasi:/dev/null
DASI_DEVICE9 dasi:/dev/null
DASI_DEVICE10 dasi:/dev/null
DASI_DEVICE11 dasi:/dev/null
DASI_DEVICE12 dasi:/dev/null

ETMS_DEVICE etms:/dev/null

TCW_URL 127.0.0.0:3001
```

Appendix A. Acronyms

AP	Application Processor
ATC	Air Traffic Control
AViD	Airspace Visualization Display
DataComm	Data Communications
DYSIM	Dynamic Simulation
EDDS	En Route Data Distribution System
EIB	ERAM in a Box
ERAM	En Route Automation Modernization
FAA	Federal Aviation Administration
GSGT	Graphic Simulation Generation Tool
JRE	Java Runtime Environment
NAS	National Air Space
NUNIO	Networked Universal Input/Output
OS	Operating System
SDRR	Simulation Driver Radar Recorder
SP	Serial Processor
STARS	Standard Terminal Automation Replacement System
TBFM	Time Based Flow Management
TDLS	Terminal Data Link Service
TFDM	Terminal Flight Data Management
TFMS	Traffic Flow Management System
WJHTC	William J. Hughes Technical Center