SN SDK Setup And Installation

Introduction

This document was created to assist in setup and installation of custom applications for Red Lion hardware.

Downloads of sdk related files can be found support page at http://www.redlion.net/industrial-wireless-software-firmware under section "**SDK Application (Sixnet RAM Series)**".

Supported Build Host

Only Linux Operating Systems are supported as a build host. Any major modern Linux distribution which supports rpm files, either natively or through a translation program, should work. For example:

- Ubuntu
- Fedora
- OpenSuse

VM Installation

You can skip this step and continue on with SDK Installation if you are running a Supported Build Host above or already have Virtual Machine with Support Build Host. This step is for if you are running Microsoft Windows operating system.

VirtualBox Installation

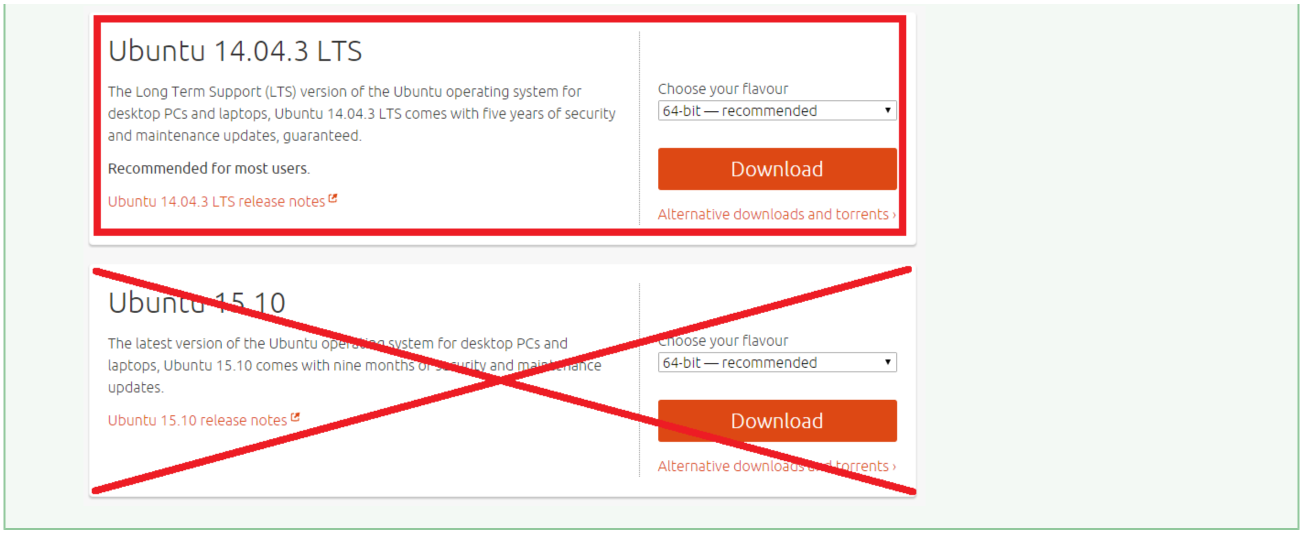
1. Download the latest version of VirtualBox from https://www.virtualbox.org/wiki/Downloads.
2. Download the Oracle VM Virtual Machine Extension Pack. The Extension Pack is for USB connections.
3. After installer has been downloaded, run the installer and install VirturalBox.
4. Once VirtualBox has installed, install the extension pack.

For details and help on VirtualBox installation, please visit https://www.virtualbox.org/manual/ch01.html

Ubuntu Installation

1. Download Ubuntu 64bit from http://www.ubuntu.com/download/desktop.

> We recommend to download the latest **long term support** version. This may not be the latest version available, but will be most stable and reliable.



Ubuntu 14.04.3 LTS

The Long Term Support (LTS) version of the Ubuntu operating system for desktop PCs and laptops, Ubuntu 14.04.3 LTS comes with five years of security and maintenance updates, guaranteed.

Recommended for most users.

Ubuntu 14.04.3 LTS release notes

Choose your flavour: 64-bit — recommended

~~Ubuntu 15.10~~

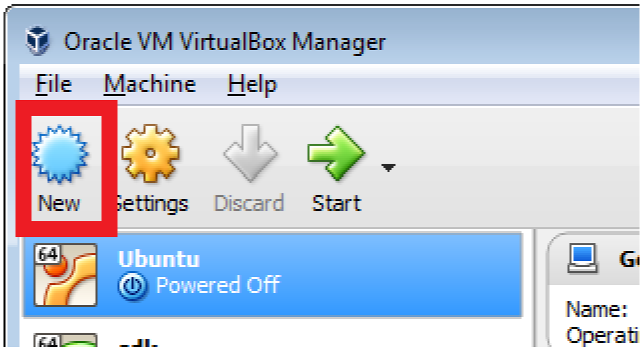
~~The latest version of the Ubuntu operating system for desktop PCs and laptops, Ubuntu 15.10 comes with nine months of security and maintenance updates.~~

~~Ubuntu 15.10 release notes~~

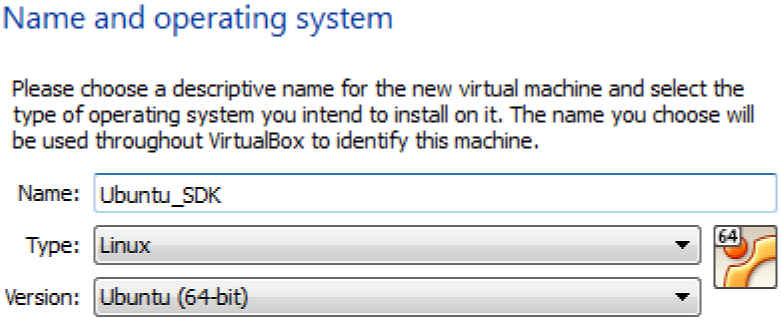
~~Choose your flavour: 64-bit — recommended~~

Download
Alternative downloads and torrents ›
Download
Alternative downloads and torrents ›

2. Run VirtualBox that you have installed in earlier steps.
3. Click **New**



4. Enter your machine name. For example "Ubuntu_SDK". Select **Linux** from *Type.* Select **Ubuntu (64-bit)** from *Version*. Click **Next**.



Name and operating system

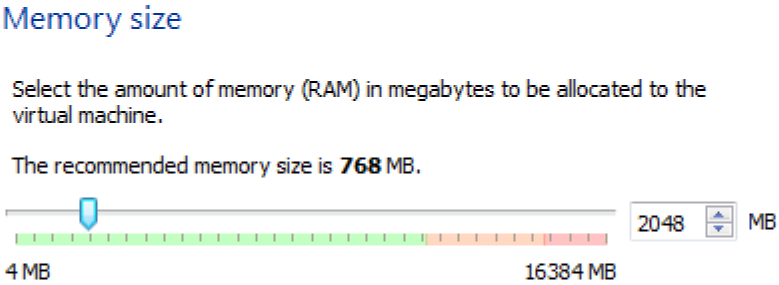
Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name: Ubuntu_SDK

Type: Linux

Version: Ubuntu (64-bit)

5. Select how much of memory you virtual machine will have. We recommend 2GB of memory. Click **Next**.



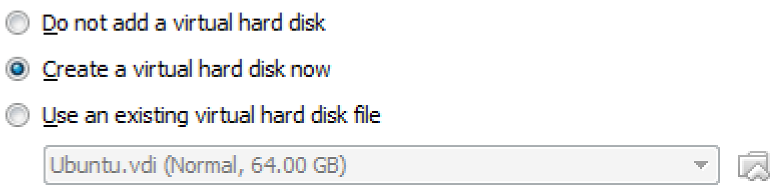
Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **768** MB.

2048 MB

6. Select **Create a virtual hard disk now** for Hard Disk. Click **Create**.



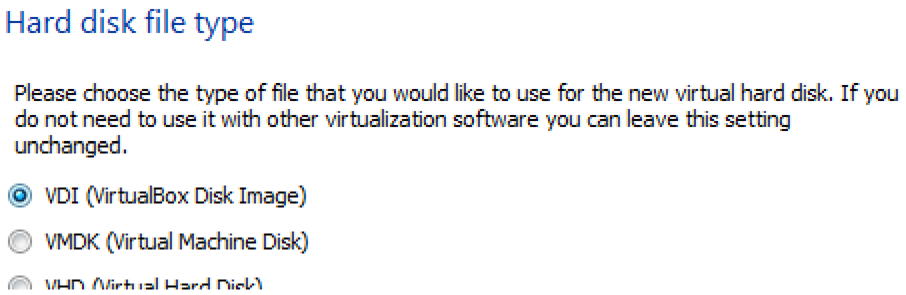
○ Do not add a virtual hard disk

● Create a virtual hard disk now

○ Use an existing virtual hard disk file

Ubuntu.vdi (Normal, 64.00 GB)

7. For *Hard dist file type*, no changes are needed, just click **Next**.



Hard disk file type

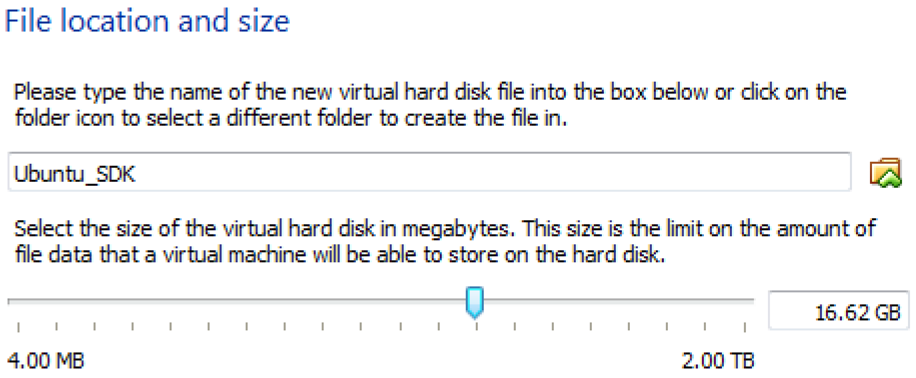
Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.

● VDI (VirtualBox Disk Image)

○ VMDK (Virtual Machine Disk)

○ VHD (Virtual Hard Disk)

8. For Storage on physical disk, leave selected as **Dynamically allocated**. Click **Next**.
9. Select where you would like your files to be save. By default they are saved in C:\Users\<UserName>\VirtualBox VMs, Select at least 8GB of HDD space. We recommend 16GB. Click **Create**.



File location and size

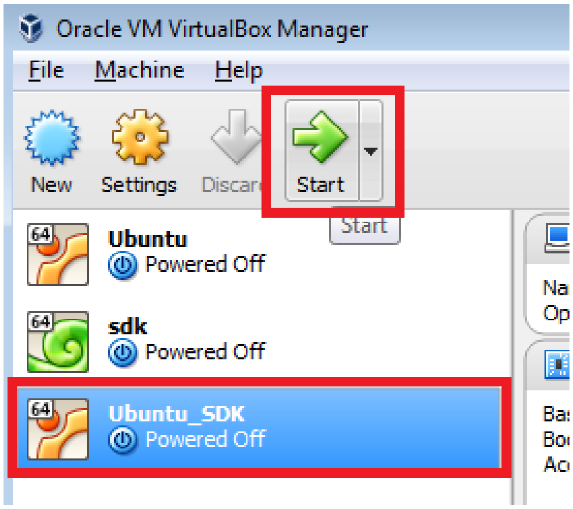
Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.

Ubuntu_SDK

Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.

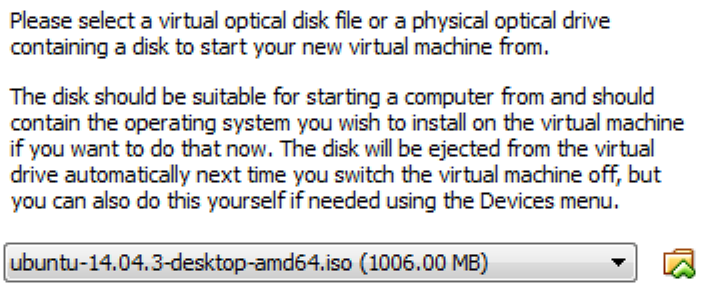
16.62 GB

10. Now you have Virtual Machine created. We are now ready to install Ubuntu onto Virtual Machine. From VirtualBox Manager, select newly created Virtual Machine and click **Start**.



11. Select start up disk by clicking

to browse for the Ubuntu ISO that you have downloaded in step 1. Click **Start**.

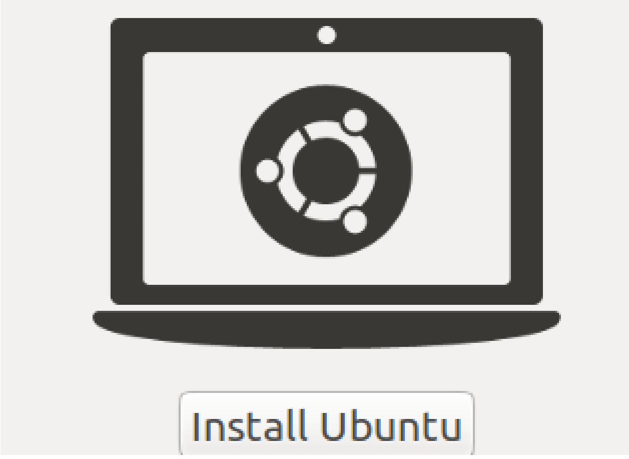


Please select a virtual optical disk file or a physical optical drive containing a disk to start your new virtual machine from.

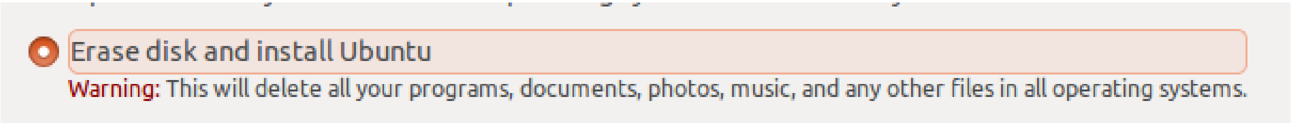
The disk should be suitable for starting a computer from and should contain the operating system you wish to install on the virtual machine if you want to do that now. The disk will be ejected from the virtual drive automatically next time you switch the virtual machine off, but you can also do this yourself if needed using the Devices menu.

ubuntu-14.04.3-desktop-amd64.iso (1006.00 MB)

12. Ubuntu will start booting. On Ubuntu *Welcome* screen, Click **Install Ubuntu**.



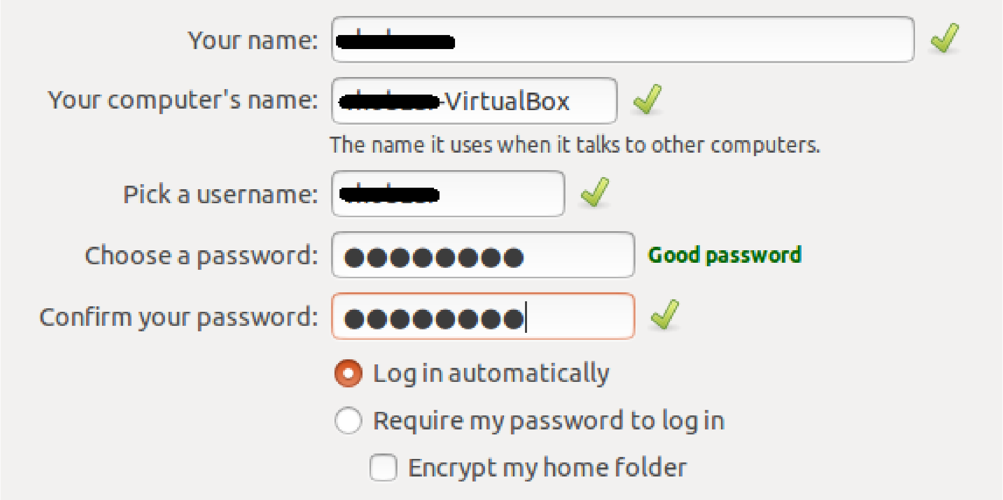
13. On *Preparing to install Ubuntu* screen, click **Continue**.
14. Select **Erase dist and install Ubuntu** as installation type and click **Install Now**.



● Erase disk and install Ubuntu

Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.

15. On *Write the changes to disk?* warning message, click **Continue**.
16. Select your timezone and click **Continue**.
17. Select your keyboard layout and click **Continue**.
18. Enter your name. Select a username. Enter a password. Select login option and click **Continue.**



Your name: [redacted] ✓

Your computer's name: [redacted]-VirtualBox ✓

The name it uses when it talks to other computers.

Pick a username: [redacted] ✓

Choose a password: ●●●●●●●● Good password

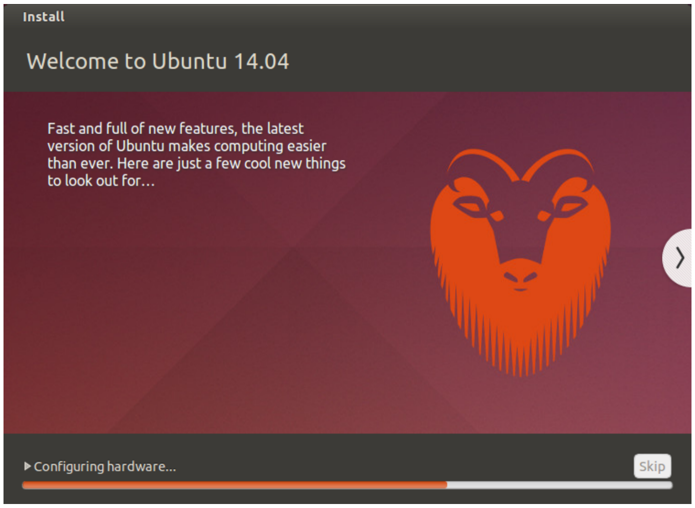
Confirm your password: ●●●●●●●● ✓

● Log in automatically

○ Require my password to log in

☐ Encrypt my home folder

19. Installation is now in progress. This may take few minutes.

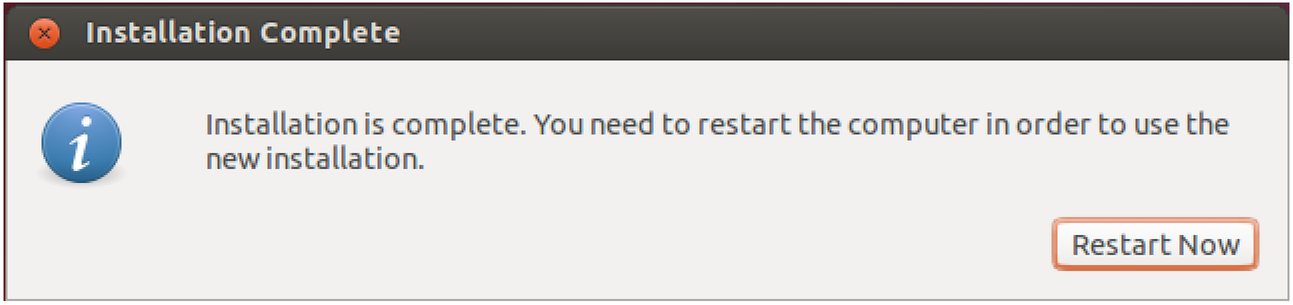


Welcome to Ubuntu 14.04

Fast and full of new features, the latest version of Ubuntu makes computing easier than ever. Here are just a few cool new things to look out for...

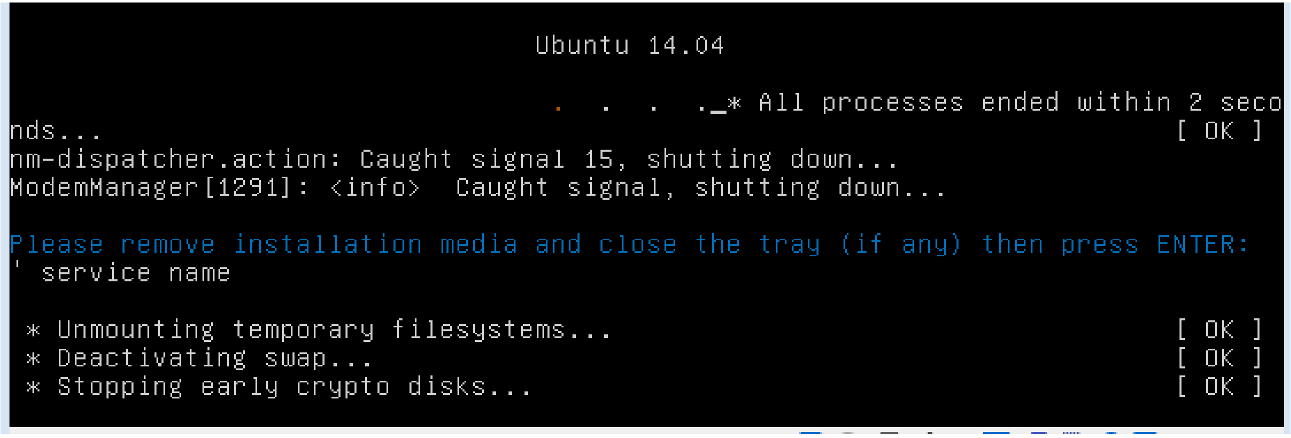
Configuring hardware...

20. Once installation is complete, click **Restart Now**.



Installation is complete. You need to restart the computer in order to use the new installation.

21. After restart, on the screen below, press **<Enter>** key.



```
                                 Ubuntu 14.04

                                  .  .  .  ._* All processes ended within 2 seco
nds...                                                                   [ OK ]
nm-dispatcher.action: Caught signal 15, shutting down...
ModemManager[1291]: <info>  Caught signal, shutting down...

Please remove installation media and close the tray (if any) then press ENTER:
' service name

 * Unmounting temporary filesystems...                                   [ OK ]
 * Deactivating swap...                                                  [ OK ]
 * Stopping early crypto disks...                                        [ OK ]
```

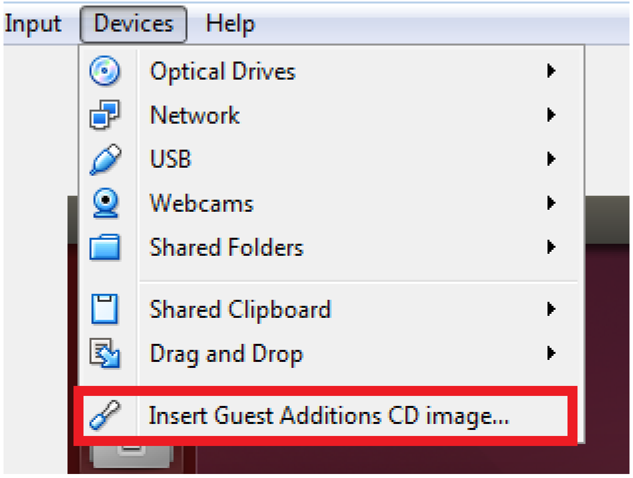
Ubuntu Setup

Screen resolution adjustment

1. On you first run, you may notice that your screen resolution is 640x480, even though you have changed the preferences to be much bigger. You need to run the following command:

```
sudo apt-get install virtualbox-guest-utils virtualbox-guest-dkms
```

2. From VirtualBox window go to **Devices**>**Insert Guest Additions CD image...**



3. Click Run to install VirtualBox Guest Additions
4. Restart Ubuntu
5. Now Ubuntu screen will adjust to VirtualBox screen size

Top

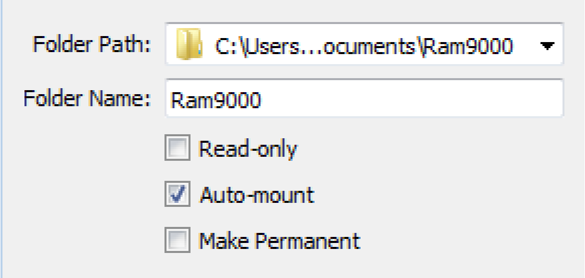
Setting up shared folders

Following steps are to setup a shared folder on Windows OS that you could access from Ubuntu. This is necessary to be able to transfer files to a from Ubuntu.

1. In Ubuntu terminal run the following command:

   ```
   sudo sed -i 's/\(vboxsf.*:\)$/\1<myuser>/' /etc/group
   ```

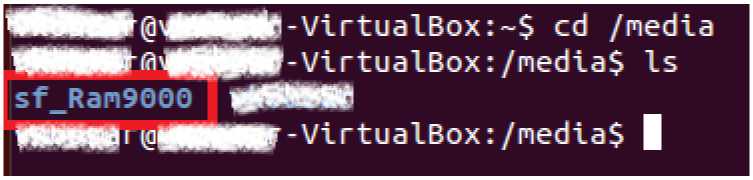
   where <myuser> is your username
2. From VirtualBox window, go to **Machine**>**Settings** and select **Shared Folders**
3. Click on Add Folder
   icon.
4. Select a folder to be shared. Give that folder a name as it will appear in Ubuntu. Select **Auto-mount**. Click **OK**

   Folder Path: C:\Users…ocuments\Ram9000
   Folder Name: Ram9000
   ☐ Read-only
   ☑ Auto-mount
   ☐ Make Permanent
5. Click **OK** to close setting window.
6. Restart Ubuntu
7. To verify that shared folder has been added, run the following commands:

   ```
   cd /media
   ls
   ```

   You should now see a folder mount with name sf_<yourSharedFolderName>



```
[redacted]r@v[redacted]-VirtualBox:~$ cd /media
[redacted]r@v[redacted]-VirtualBox:/media$ ls
sf_Ram9000  [redacted]
[redacted]r@[redacted]-VirtualBox:/media$
```

Top

SDK Installation

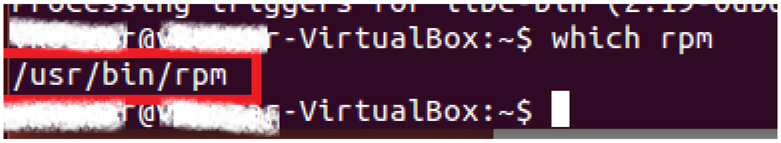
Requirements

1. A relatively modern Linux distribution
2. The rpm command installed on your distribution. To check, in a terminal, execute:

   ```
   which rpm
   ```

   If you get a reply like:

   ```
   /usr/bin/rpm
   ```

   ```
   [redacted]r@v[redacted]r-VirtualBox:~$ which rpm
   /usr/bin/rpm
   [redacted]r@v[redacted]-VirtualBox:~$
   ```

   then you have it installed. If not, please consult your distribution’s documentation on how to install it.
   On Ubuntu run the following command to install rpm:

   ```
   sudo apt-get install rpm
   ```

3. At least 2.5 GB of free space.

ELDK Toolchain Installation

Download

Software development for the SN series products utilizes the DENX Embedded Linux Development Kit (ELDK), version 4.2, for ARM. Full documentation for this toolchain can be accessed at: http://www.denx.de/wiki/DULG/ELDK

But in order to not swamp DENX’s bandwidth, however, Red Lion maintains a mirror of the toolchains we use at public software page http://www.redlion.net/industrial-wireless-software-firmware.

File name: **`arm-2008-11-24.iso`**

64bit Host Installation Prerequisite

As the ELDK is compiled for 32-bit host systems, a compatibility layer is required on 64-bit host systems.

For Ubuntu, run the following to install ia32.libs

```
sudo apt-get install libc6:i386
sudo -i
cd /etc/apt/sources.list.d
echo "deb http://old-releases.ubuntu.com/ubuntu/ raring main restricted
universe multiverse" >ia32-libs-raring.list
apt-get update
apt-get install ia32-libs
rm /etc/apt/sources.list.d/ia32-libs-raring.list
apt-get update
exit
sudo apt-get install gcc-multilib
```

For Fedora, the following should be enough

```
sudo yum -y install glibc.i686 zlib.i686
```

Installation

Toolchain Target Directory
By default, the Makefile in the bootstrap build environment will expect the toolchain to be installed into

```
/opt/eldk/4.2
```

If you install to this location, you won’t have to change anything in the example. If you do not, you will have to change the TOOLCHAIN_PATH export in Makefile.

Either burn the .iso to a CD with your favorite application or loopback mount the .iso as follows:

```
sudo mkdir /media/iso
sudo mount -o loop arm-2008-11-24.iso /media/iso/
```

Then run the installer, substituting values for the placeholders (in < >’s) as appropriate:

```
mkdir <target directory>
/media/iso/install -d <target directory>
```

Once installed, unmount the .iso and remove the directory we created:

```
umount /media/iso
rmdir /media/iso
```

Top

SN-SDK Library Installation

The libraries in the supplemental tarball provide headers and linkable shared objects that reflect what is *actually* on-device to ease third-party development of applications which run on the SN series hardware. They are supplied as a tarball which is designed to be an “overlay” over the toolchain.

File Name: **`snsdk.4.23.tgz`**

Installation

Untar the library tarball over the installed target directory.

> Note that **<target directory>** here should be the same as where you installed the ELDK.

```
tar -zxf snsdk.4.23.tgz -C <target directory>
```

Top

Bootstrap Installation

The bootstrap build environment is provided as a convenience to users who wish to get up and running quickly. It contains a sandbox folder with scripts for streamlining compiling and packaging a program, and a folder of reference examples/documentation.

For more information on how to configure/build/install an application using SN-SDK please go to SN-SDK User Guide

File name: **`snsdk_bootstrap_v4.tgz`**

Installation

Untar the tarball to some convenient <working directory> in which you would like to develop code.

```
tar -zxf snsdk_bootstrap_v4.tgz // Current Directory

// OR

tar -zxf snsdk_bootstrap_v4.tgz -C <working_directory> // to some
<working_directory>
```

Top

SN-SDK User Guide

- Example Program

This user guide uses an Example program. This code to this program could be found here and also it is included in the bootstrap.

SDK Application Naming
Please Note: With current firmware, you can use Expert Mode in the Web UI to stop/start your application and have access to a configuration file. To do this, your application must be named "sdk."

Building the Program

The example program is available as native C source code in the bootstrap environment, along with the necessary Makefile(s) and directory structure required to build a package for installation on your RAM series product.

To build the example program (iodbEx), simply navigate to the sandbox directory in the root of your bootstrap environment installation, and:

1. Place all of your source code in the sandbox/source folder. By default, all .c files will be included in the compile.
   Run the following command to copy Example program to source forlder.

   ```
    cp ../reference/iodbEx.c source
   ```

2. Configure the build environment for your program and create accessory files by executing the following from the sandbox directory in terminal.

   ```
   ./configure.sh
   ```

   This script will create a Makefile in the sandbox directory, as well as sdk.init , sdk.conf, and install.sh in the accessory folder.
3. Compile your program:

   ```
   make
   make install
   ```

   This will create a **sdk.zip** in the packages folder. Now you can use this zip file to install your (Example) program unto Ram device.

Now that you have successfully compiled the example program (make) and built a package for installation on your RAM series product (make install), you need to install the package.

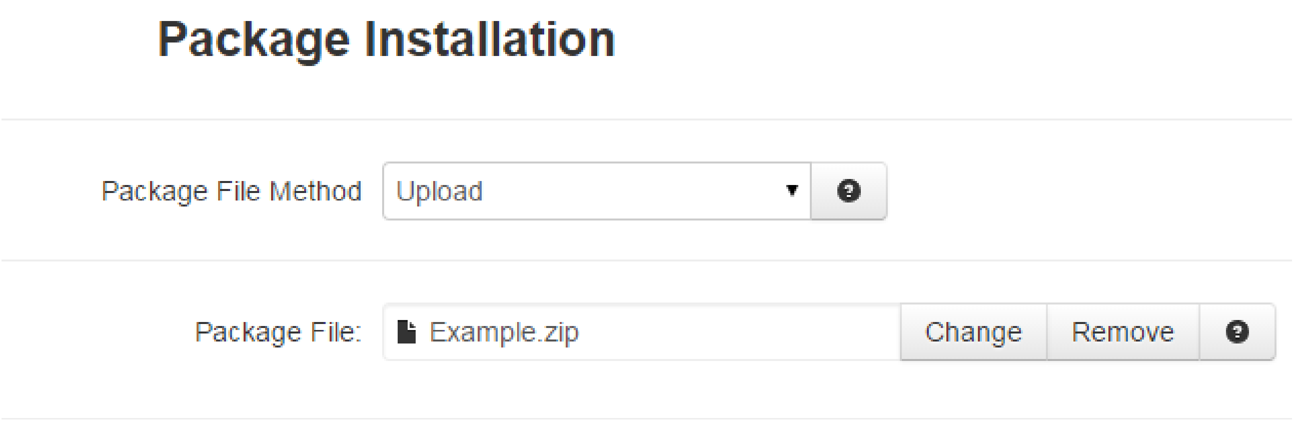
RAM series package files are simple .zip archives, containing full directory structures that will correspond to the installation location on your RAM series product. Further, there is a special (optional) file that can control more advanced behavior during package installation.

Installing the Example Program

There are three ways to install the example program package:

Product On-Board Web Interface Installation

1. In product on-board web interface, to got **Admin**>**Package Installation**
2. Select package method - Upload the zip file or get it from the SD-Card (if equipped).
3. Select your zip package

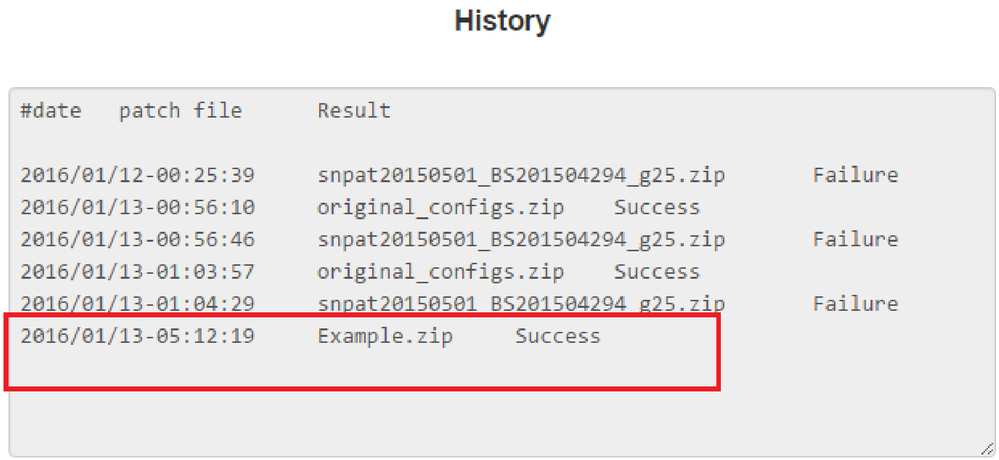


Package Installation

Package File Method: Upload

Package File: Example.zip

4. Click "Install" at the bottom of the page
5. Click "OK" to confirm
6. Verify in the History block that the package has installed successfuly



History

```
#date   patch file      Result

2016/01/12-00:25:39     snpat20150501_BS201504294_g25.zip       Failure
2016/01/13-00:56:10     original_configs.zip    Success
2016/01/13-00:56:46     snpat20150501_BS201504294_g25.zip       Failure
2016/01/13-01:03:57     original_configs.zip    Success
2016/01/13-01:04:29     snpat20150501_BS201504294_g25.zip       Failure
2016/01/13-05:12:19     Example.zip     Success
```

Sixview Manager Installation

The same .zip file you created for package installation through the GUI can be uploaded to a Sixview Manager server for remote distribution to a unit. Simply create a job and add the package file to it. See Sixview Manager documentation for more details on uploading Patch files and creating jobs.

Command Line Interface

> Command Line Interface is not recommended for customer user.

Upload to /tmp via TFTP, FTP, SCP or ZModem and install via:

```
[root@SNgateway-v3_09 tmp]# snupdate sdk.zip
snupdate v1.15
** Package contents appear valid.
** Running included installation script.
Archive: /tmp/sdk.zip
inflating: /usr/local/bin/sdk
Install success
```

Additional Details

Libraries

There is a limited space available for your application, and it is advised to use included libraries when possible. Please check this list for libraries that are included on the device before you install your own copies of same libraries.

configure.sh

The configure script copies default files for your program and modifies install.sh if you specify whether your program should start when the device boots and/or persist through reflashing. To bypass this script, copy the default files needed for package installation like so:

```
cp defaults/Makefile .
cp defaults/sdk.init accessory/
cp defaults/sdk.conf accessory/
cp defaults/install.sh accessory/
```

sdk.conf

sdk.conf is a configuration file that you can edit through the GAU Sub-systems menu in Expert Mode. The configure script creates a default in the accessory folder. You can rewrite this file to be used by your program. On the device, this file will be located here:

```
/etc/jbm/sdk.conf
```

sdk.conf is format independent and it is up to the SDK user to implement a compatible reader for any selected content.

See iodbTemplate.c in the reference directory for example usage of the daemon() function to add an application debug mode. Daemonizing will redirect all standard output and errors to /dev/ null as is the default behavior based on the sdk.init script. In the example, this is overridden by the -d command-line option.

Controlling Package Installations

Let’s start by examining the contents of the example package archive (sdk.zip):

Archive: iodbEx.zip

```
Length   Date   Time  Name
---------  ----------  -----  ----
  0 2012-04-04  18:23  etc/
  0  2012-04-04  18:23  etc/jbm/
    27  2012-04-04  18:23  etc/jbm/sdk.conf
    27  2012-04-04  18:23  etc/jbm/sdk.conf.orig
  0  2012-04-04  18:23  etc/rc.d/
  0  2012-04-04  18:23  etc/rc.d/init.d/
   964  2012-04-04  18:23  etc/rc.d/init.d/sdk
     1159  2012-04-04  18:23  install.sh
  0  2012-04-04  18:23  usr/
  0  2012-04-04  18:23  usr/local/
  0  2012-04-04  18:23  usr/local/bin/
 11096  2012-04-04  18:23  usr/local/bin/sdk
---------  -------
 13273  12 files
```

Notice the file in the package manifest named install.sh. This bash script will be executed, if present, before any package installation is executed. The sample install.sh file included in the bootstrap environment is a straightforward example that simply extracts the contents of the archive to the RAM series product and logs a message to syslog.

Without this special file, the example program would simply be extracted from the archive by the package installer, with no additional logic.

Two lines are key in allowing advanced functionality for the program.

The following line enables the init script for the program meaning it will start whenever the device boots. Uncomment to enable this functionality.

```
#chkconfig sdk on
```

This line copies the package file into a special directory causing it to be reinstalled next time the device is flashed. Uncomment to enable.

```
#cp -f $FILE_NAME /storage/sdk/install/
```

WARNING: Configuring an untested package to install every time, or on bootup, could render your device inoperable. The SDK is a powerful tool. Please test extensively before applying permanent changes.

Also, see additional snupdate documentation for more information on creating package .zip files.

install.sh

install.sh

```
#!/bin/sh

# Check that /etc/version exists. This pretty much only true on JBM/SN/Red
Lion devices
if [ ! -e "/etc/version" ]; then
    echo "Unit does not appear to be a JBM/Sixnet/Red Lion Controls unit,
exiting."
    exit 1
fi

# The only argument we are passed is the file name of the package
FILE_NAME=$1

# Make sure we can find that file before we continue
if [ ! -e "$FILE_NAME" ]; then
    echo "Cannot find package file $FILE_NAME"
    exit 2
fi

# Extract the rest of the contents of the package
# -d             The destination directory for the unpacked contents
# -o             Overwrite existing files without prompting (use this!)
# $FILE_NAME     The file name of the package to unzip
# -x install.sh  Exclude install.sh when extracting since that has already
been unpacked
unzip -d / -o $FILE_NAME -x install.sh

# Uncomment to turn on autostart of sdk process at boot time
#chkconfig sdk on

# Uncomment the following line to allow the program to persist across
reflashing
#cp -f $FILE_NAME /storage/sdk/install/

# Uncomment the following lines to add SDK tab to navigation bar
#if ! grep --quiet "^SDK" /home/httpd/jbmconfig/txt/customTabs.txt; then
#    echo 'SDK,sdk.html' >> /home/httpd/jbmconfig/txt/customTabs.txt
#else
#    sed -i 's/^SDK.*$/SDK,sdk.html/'
/home/httpd/jbmcofig/txt/customTabs.txt
#fi

# Now that the contents are extracted, we can do any other action the
package needs to perform
logger -t $FILE_NAME "Initial installation..."


exit 0
```

Example Application Persistence

In order to allow your application(s) to persist across firmware upgrades, they must be stored in a special filesystem and format on the RAM series product. The simplest way to do this is to move your tested installation package to the /storage/sdk/install directory on your RAM series product. This can be accomplished with:

```
cp -f /tmp/iodbEx.zip /storage/sdk/install/
```

This has been added to the install.sh script, and is disabled by default.

More on application persistence, please see Package Preservation

GAU Interaction with SDK Application

The SDK subsystem can be managed through the GAU in expert mode. This is available by going to Advanced > Expert Mode > Configure Sub-systems. Selecting "SDK" from the drop-down menu will display an editable configuration file and buttons for starting and stopping the application.

If the sdk application uses the configuration file, it can be modified here. A copy of the original file is created by the bootstrap when packaging the program, and can be restored by clicking the "Default" button.

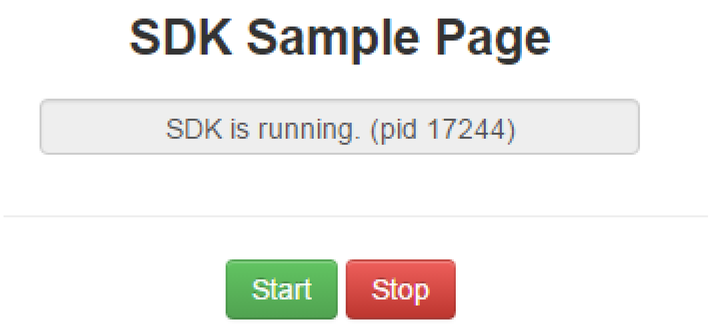
> This section applies if you named you application "sdk" only.

Alternatively you may create your own web page and add it as extension tab/page to device GUI interface.

- Details on setting up your own interface to be served by our web server, see Web UI Integration
- For version 4.23 see Add Custom Tabs to Navigation
- For version 4.24 and later see GAU Custom Extensions

Sample custom SDK page

An sample SDK page is included in the bootstrap in /reference/home. This "home" directory contains the entire directory structure and files needed to install SDK sample page. On this page you have two controls, start and stop your application.



SDK Sample Page

SDK is running. (pid 17244)

Step to enable SDK Sample Page

1. run copy command to copy home directory to accessory directory

```
# assuming you are in sandbox
cp -rf ../reference/home/ accessory/
```

2. Uncomment the following lines in the install.sh

```
# Uncomment the following lines to add SDK tab to navigation bar
if ! grep --quiet "^SDK" /home/httpd/jbmconfig/txt/customTabs.txt;
then
    echo 'SDK,sdk.html' >> /home/httpd/jbmconfig/txt/customTabs.txt
else
    sed -i 's/^SDK.*$/SDK,sdk.html/'
/home/httpd/jbmcofig/txt/customTabs.txt
fi
```

3. To include home directory structure, run

```
make installTab
```

Example Program

This code is written to give an example or two on how to access the SN I/O database from external OEM applications.

Discretes are bitpacked. 8 bits per byte. Floats are treated as floats (4 bytes), not doubles. Longs are treated as longs (4 bytes).

iodbEx.c

```
/*
 * Example implementation of iodb functions
 *
 * Copyright 2016 Red Lion Controls, Inc.  All rights reserved.
 */
/* -*- mode:c;  c-basic-offset: 4; tab-width: 8; -*- vi: set sw=4 ts=8:*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <iodb.h>

/* Types */
#define INT16   short
#define INT32   long
#define UINT16  unsigned short
#define UINT32  unsigned long
#define ULONG   unsigned long
#define BYTE    unsigned char
#define BOOL    int

/* IODB Types */
#define ANALOG_IN       0
#define ANALOG_OUT      1
#define DISCRETE_IN     10
#define DISCRETE_OUT    11
```

```
#define LONG_IN         20
#define LONG_OUT        21
#define FLOAT_IN        22
#define FLOAT_OUT       23


int main()
{
    int i;
    UINT16 addr;         /* Starting Register */
    UINT16 regs;         /* Number of Register */
    UINT16 datatype;     /* Type of I/O */
    UINT16 typesize;
    UINT16 typenum;
    INT16 ecode;
    INT16 a_value;
    INT16 analog_values[8];
    INT16 discrete_values;

    /*
     * Write 1 analog value to analog output register 0.
     * Then read the value back in...
     */
    addr = 0;
    regs = 1;
    a_value = 37;
    printf("\nWriting 1 analog, Reading 1 back...\n");
    printf("a_value: %d\n", a_value);
    ecode = IODBWrite(ANALOG_OUT, addr, regs, (void *)&a_value, NULL);
    a_value = 0;
    ecode = IODBRead(ANALOG_OUT, addr, regs, (void *)&a_value, NULL);
    printf("a_value (returned): %d\n", a_value);

    /*
     * Write 3 analog values to 3 consecutive registers,
     * starting at analog output register 1.
     * Then read the values back from register 0.
     */
    addr = 1;
    regs = 3;
    analog_values[0] = 11;
    analog_values[1] = 22;
    analog_values[2] = 33;
    printf("\nWriting 3 analogs, Reading 4 back...\n");
    for (i = 0; i < regs; i++) {
        printf("analog_values[%d]: %d\n", i, analog_values[i]);
    }
    ecode = IODBWrite(ANALOG_OUT, addr, regs, (void *)&analog_values,
NULL);

    memset((void *)&analog_values, '0', sizeof(INT16) * 8);
    addr = 0;
    regs = 4;
    ecode = IODBRead(ANALOG_OUT, addr, regs, (void *)&analog_values, NULL);
    for (i = 0; i < regs; i++) {
        printf("analog_values[%d]: %d\n", i, analog_values[i]);
    }

    /*
     * Write 1 discrete value to discrete output register 0.
     * Then read the value back in...
     */
    addr = 0;
    regs = 1;
    discrete_values = 1;  /* Turning discrete ON */
    printf("\nWriting 1 discrete, Reading 1 back...\n");
    printf("discrete_values: %d\n", discrete_values);
    ecode = IODBWrite(DISCRETE_OUT, addr, regs, (void *)&discrete_values,
NULL);
    discrete_values = 0;
    ecode = IODBRead(DISCRETE_OUT, addr, regs, (void *)&discrete_values,
NULL);
    printf("discrete_values (returned): %d\n", discrete_values);

    /*
     * Write 4 discrete value to discrete output register 4.
     * Then read 8 discretes back, starting at register 0...
     *    D0 = ON  (Previous Write)
     *    D4 = ON
     *    D6 = ON
     *    If nothing else is ON, should return 81.
     */
    addr = 4;
    regs = 4;
    discrete_values = 5;  /* D0 & D2 are on, D1 & D3 are off */
    printf("\nWriting 4 discretes, Reading 8 back...\n");
    printf("discrete_values: %d\n", discrete_values);
    ecode = IODBWrite(DISCRETE_OUT, addr, regs, (void *)&discrete_values,
NULL);
    discrete_values = 0;
    addr = 0;
    regs = 8;
    ecode = IODBRead(DISCRETE_OUT, addr, regs, (void *)&discrete_values,
NULL);
    printf("discrete_values (returned): %d\n", discrete_values);


    /*
     * Reset Digital outs to 0
     *
     */
    addr = 0;
    regs = 3;
    discrete_values = 0;
    ecode = IODBWrite(DISCRETE_OUT, addr, regs, (void *)&discrete_values,
NULL);
```

```
    /*
     * Print current SDK Application time to screen every 5 seconds while
app is running.
     */
    int M,D,Y,h,m,s;
    FILE *fp;
    while(1)
    {
        // Read current date/time
        M = D = Y = h = m = s = 0;

        IODBRead(ANALOG_OUT, 1010, 1, (void *)&Y, NULL); // Year
        IODBRead(ANALOG_OUT, 1011, 1, (void *)&M, NULL); // Month
        IODBRead(ANALOG_OUT, 1012, 1, (void *)&D, NULL); // Day
        IODBRead(ANALOG_OUT, 1015, 1, (void *)&h, NULL); // Hour
        IODBRead(ANALOG_OUT, 1016, 1, (void *)&m, NULL); // Minute
        IODBRead(ANALOG_OUT, 1017, 1, (void *)&s, NULL); // Second

        // Write time to local file
        fp = fopen("/tmp/sdk.txt", "w");
        fprintf(fp, "SDK Application last run at: %d/%d/%d %d:%d:%d
\n",M,D,Y,h,m,s);
        printf("SDK Application last run at: %d/%d/%d %d:%d:%d
\n",M,D,Y,h,m,s);
        fclose(fp);

        // Stop the application if first word in the config file says
"Stop"
        char buff[255];
        fp = fopen("/etc/jbm/sdk.conf", "r");

        fscanf(fp, "%s", buff);
        if(strcmp(buff, "Stop") == 0)
        {
            break;
        }
        fclose(fp);

        sleep(5);
    }


    exit (0);
}
```

Top

SN SDK IODB Library Reference

IODB Functions

These library functions are used within applications to exchange I/O with a RAM Device station's I/O registers.
Upcoming is a list of the function prototypes. Each function topic includes a C example.
Refer to the supplied sample program, iodbEx.c, which uses these library function calls.

These functions are used to read and write station I/O registers.

IODBRead

Use this function to:
Read one or more consecutive values from the I/O database based on type number and address.

Arguments:

Argument	In/Out	Definition
TypeNum	in	Register type
Addr	in	Starting address
NumRegs	in	Number of registers to read
pbuff	out	Buffer to read data into
StaName		Not used. Pass NULL.

Return Values:

Value	Definition
`ENOERROR`	on success
`EOUTOFRANGE`	if Addr+NumRegs exceeds values vailable for TypeNum
`ENOSTATION`	if Addr or TypeNum is invalid for station
`EINVALIDTYPE`	if TypeNum is invalid
`EDBERROR`	if database cannot be opened.

Prototype:

```
IODBerr  IODBRead (
USHORT  TypeNum,
USHORT  Addr,
USHORT  NumRegs,
void  *pbuff,
const char  *StaName
);
```

Examples:
iodbEx.c.

Top

IODBWrite

Use this function to:
Write one or more consecutive I/O values into the I/O database based on type number and address.

Arguments:

Argument	In/Out	Defnition
TypeNum	in	Register type
Addr	in	Starting address
NumRegs	in	Number of registers to write
pbuff	in	Buffer containing data to write
StaName	in	Not used. Pass NULL.

Return Values:

Value	Definition
ENOERROR	on success
EOUTOFRANGE	if Addr+NumRegs exceeds values available for TypeNum
ENOSTATION	if Addr or TypeNum is invalid for station
EINVALIDTYPE	if TypeNum is invalid
EDBERROR	if cannot open database for write.

Prototype:

```
IODBerr IODBWrite (
USHORT   TypeNum,
USHORT   Addr,
USHORT   NumRegs,
void   *pbuff,
const char  *StaName
);
```

Examples:
iodbEx.c.

Top

IODBWriteMask

Use this function to:
Write selected I/O values into I/O database.
The bit mask is used to indicate whether an I/O register is to be written to or not. A 0 indicates no, a 1 indicates yes.

Arguments:

Argument	In/Out	Definition
TypeNum	in	Register type
Addr	in	Starting address
NumRegs	in	Number of registers to write
pbuff	in	Buffer containing data to write
pMask	in	Controls which values to write
StaName	in	Not used. Pass NULL.

pMask specifies the write mask (a 1 indicates the point will be updated, and a 0 indicates the point will remain unchanged). The bit order of the mask is as follows: the least-significant bit of the first byte of pMask corresponds to the register at address Addr. The bit positions continue in ascending order to the most-significant bit and then continues on to the least-significant-bit of the next byte.

Return Values:

Value	Definition
ENOERROR	on success
EOUTOFRANGE	if Addr+NumRegs exceeds values vailable for TypeNum
ENOSTATION	if Addr or TypeNum is invalid for station
EINVALIDTYPE	if TypeNum is invalid
EDBERROR	if cannot open database for write

Prototype:

```
IODBerr IODBWriteMask (
uint16_t  TypeNum,
uint16_t  Addr,
uint16_t  NumRegs,
void   *pbuff,
uint8_t  *pMask,
const char  *StaName
);
```

Top

IODBVersion

Use this function to:
Retrieve the current version of the DLL running for this library.

Return Values:

Library version. The most significant byte is the major version. The least signficant byte is the minor version.

Prototype:

```
USHORT IODBVersion (void);
```

Top

File List

Here is a list of all documented files with brief descriptions:

iodb.h File Reference

Declaration of SIXNET IODB functions.

```
#include "utypes.h"
#include <time.h>
```

Defines

```
#define IODB_VERSION 0x0100
#define EDBERROR -1
#define ENOERROR 0
#define EOUTOFRANGE 1
#define EINVALIDTYPE 2
#define ENOSTATION 3
#define EFILEACCESS 4
```

```
#define EFILENOTFOUND 5
#define EIODBNOTFOUND 6
#define EOUTOFMEMORY 7
#define ETASKERROR 8
#define ENONSIXNETFILE 9
#define ETAGNOTFOUND 10
#define EDUPLICATETAG 11
#define EBUFFERSIZE 12
#define ENOTANALOGTYPE 13
#define EREGISTERNOTFOUND 14
#define EEXCEEDEDMAXTAGS 15
#define NOFORMAT 1
#define SFORMAT 2
#define UFORMAT 3
#define LFORMAT 4
#define ULFORMAT 5
#define RESERVEDtype 0
#define ANALOGtype 1
#define DISCRETEtype 2
#define BYTEtype 3
#define LONGtype 4
#define FLOATtype 5
#define DOUBLEtype 6
#define USERtype 7
#define READSCAN 0
#define WRITESCAN 1
#define WRITEREAD 2
#define EXCEPTION 3
#define ASSIGNED 4
#define SIXTRAK 0
#define IOMUX 1
#define VERSAMUX 2
#define S60IBMN 3
#define LOCALCOMP 4
#define IODB 5
#define MISCTYPE 6
#define VERSATRAK 7
#define REMOTETRAK 8
#define ETHERTRAK 9
#define SYSPLAN_PROG 0
#define SPF_PROG 1
#define SCR_PROG 2
#define SCRADV_PROG 3
#define SCR_1131 4
#define SCR_1131_ADV 5
#define NO_LICENSE -1
#define TEMPORARY_LICENSE 0
#define DEMO_LICENSE 1
#define SINGLE_LICENSE 2
#define SITE_LICENSE 3
#define OEM_LICENSE 4
```

```
#define BYTE unsigned char
#define USHORT unsigned short
#define IODBerr unsigned short
```

Functions

```
IODBerr IODBRead (USHORT, USHORT, USHORT, void , const char )
// Read one or more consecutive values from the I/O database based on type
number and address.

IODBerr IODBWrite (USHORT, USHORT, USHORT, void , const char )
// Write one or more consecutive I/O values into the I/O database based on
type number and address.

IODBerr IODBWriteMask (USHORT, USHORT, USHORT, void , BYTE , const char )

USHORT IODBVersion (void)
// Retrieve the current version of the DLL running for this library.
```

Top

SDK How-To's

This page is devoted to quick guide on how do I do a particular thing on the Red Lion Device using SN-SDK.

How do I make changes to the system config?

Where should I save my SDK program and config files?

> A: Files that need to be persistent such as the ".conf" files should be stored in Vault. More detail on the Vault partition here.

How do I control the cellular connection?

How do I send/receive SMS messages?

How can I send email?

How do I re-flash from the command line?

> A: Copy the .jffs2 files into the /tmp directory. Then run the command "cliflash" with no arguments. It will ask you if you want to save the configuration, and afterwards proceed with re-flashing the device.

How do I read physical and virtual IO?

> A: Information and details about how to read and write IODB values can be found on SN-SDK User Guide page in the Example Program section.

How do I get system statistic/information?

How do I get GPS information?

How to execute AT Commands?

XFGLib User Guide

- Step 1 - copy config.xml
- Step 2 - Make GUI changes
- Step 3 - note changes made
- Step 4 - Prepare perl script
    - Difference between "Save" and "Apply"

- Step 5. Verify your perl script
- Step 6. Prepare install.sh

This user guide will guide you through how to use xfglib and how to implement it into your install.sh of your sdk application. Subsystem sshserver will be used as an example subsystem.

Step 1 - copy config.xml

Copy existing(before making changes) config.xml to tmp or some other directory for reference.

```
cp /home/httpd/jbmconfig/conf/config.xml /tmp/
```

If you know the name of your subsystem, you can look up it current setting with grep

```
[root@SNgateway-v4_24_BETA-28 tmp]# grep -A 10 "sshserver subsystem"
/tmp/config.xml
    <sshserver subsystem="sshserver">
        <enable>n</enable>
        <advanced>n</advanced>
        <listenaddr>0.0.0.0</listenaddr>
        <listenport>22</listenport>
        <protocol>2</protocol>
        <gracetime>90</gracetime>
        <maxstartups>10</maxstartups>
        <rootlogin>n</rootlogin>
    </sshserver>
```

Now we can see current ssh settings as they appear in config.xml.

Step 2 - Make GUI changes

Now make changes to GUI settings.

In our example go to Services>SSH/TELNET Server

- Set Enable SSH Server: Yes
- Set Show Advanced Configuration: Yes
- Set Allow Root Login: Yes
- Click Save

Step 3 - note changes made

Now that you have made changes in GUI, compare original config.xml with latest config.xml. You can view either view same section of the of the config.xml and compare it with results from step 1, or you can run diff on original and latest config.xml files.

Viewing latest config.xml

```
[root@SNgateway-v4_24_BETA-28 tmp]# grep -A 10 "sshserver subsystem"
/home/httpd/jbmconfig/conf/config.xml
    <sshserver subsystem="sshserver">
        <enable changed="1">y</enable>
        <advanced changed="1">y</advanced>
        <listenaddr>0.0.0.0</listenaddr>
        <listenport>22</listenport>
        <protocol>2</protocol>
        <gracetime>90</gracetime>
        <maxstartups>10</maxstartups>
        <rootlogin changed="1">y</rootlogin>
    </sshserver>
```

runing diff

```
[root@SNgateway-v4_24_BETA-28 tmp]# diff /tmp/config.xml
/home/httpd/jbmconfig/conf/config.xml
--- /tmp/config.xml     Tue Feb 23 03:31:45 2016
+++ /home/httpd/jbmconfig/conf/config.xml       Tue Feb 23 03:44:49 2016
@@ -573,14 +573,14 @@
         <hostport>20000</hostport>
     </snproxy>
     <sshserver subsystem="sshserver">
-        <enable>n</enable>
-        <advanced>n</advanced>
+        <enable changed="1">y</enable>
+        <advanced changed="1">y</advanced>
         <listenaddr>0.0.0.0</listenaddr>
         <listenport>22</listenport>
         <protocol>2</protocol>
         <gracetime>90</gracetime>
         <maxstartups>10</maxstartups>
-        <rootlogin>n</rootlogin>
+        <rootlogin changed="1">y</rootlogin>
     </sshserver>
```

Comparing the changes, you can see that tags `enable`, `advanced` and `rootlogin` have changes. The original values were "n", and latest are "y" These are the tags that can be used in the perl script for xfglib.

Step 4 - Prepare perl script

Details on xfglib and function structure is found in XFGLib reference documentation.

Lets say now we would like to make the following changes to ssh:

> Please note, the changes below are for demonstration purpose only.

- Listening IP Address: **10.0.0.1**
- Login Grace Time (seconds): **120**
- Maximum Concurrent Connections: **20**
- Allow Root Login: **No**

For the four fields above, looking at step 1 and 3, our tags will be as follows

- Listening IP Address: **listenaddr**
- Login Grace Time (seconds): **gracetime**
- Maximum Concurrent Connections: **maxstartups**
- Allow Root Login: **rootlogin**

Now we construct our perl scrip and populate tags.

```
require "/etc/jbm/xfglib.pl";

my %xml_settings = (
            listenaddr => "10.0.0.1",
            gracetime => "120",
            maxstartups => "20",
            rootlogin => "n"
            );
&xfg_set_multi_attribute("sshserver", \%xml_settings);

&xfg_commit("apply");
```

Save your perl script to /tmp/ folder in you sdk package something like "mySSHsettings.pl".

Difference between "Save" and "Apply"

At the end of our perl script we run the following command

```
&xfg_commit("apply");
```

which will either save or apply our changes config.xml.

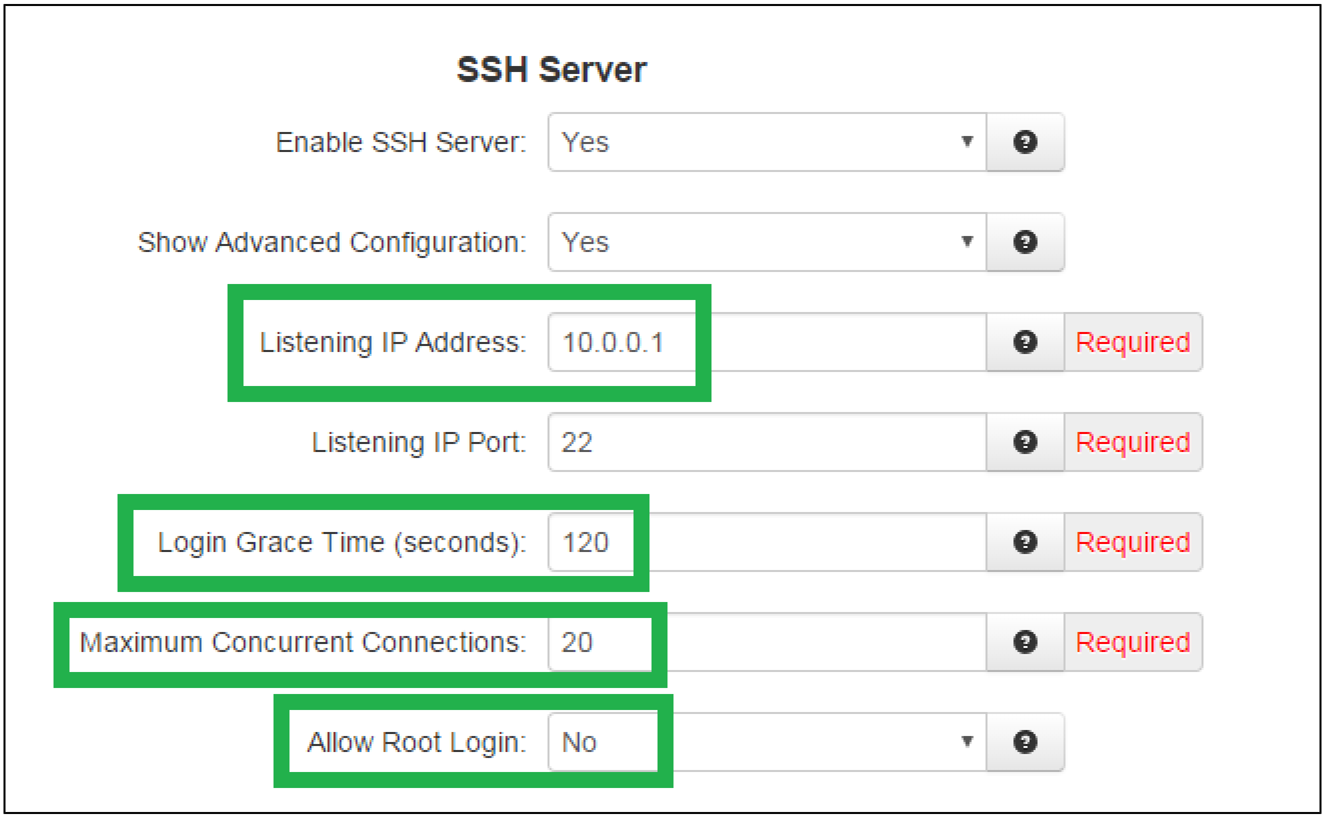
- "apply" - will save our changes to config.xml and changes will take effect immediately.
- "save" - will only save changes to config.xml. Changes will take effect after next reboot.

Step 5. Verify your perl script

Once you have make all change to your perl script, copy mySSHsettings.pl to your device and run the following command:

```
perl /tmp/mySSHsettings.pl
```

After script finished executing. go to Services>SSH/TELNET Server in you GUI browser and verity that changes are make correctly.



SSH Server

Enable SSH Server: Yes

Show Advanced Configuration: Yes

Listening IP Address: 10.0.0.1

Listening IP Port: 22

Login Grace Time (seconds): 120

Maximum Concurrent Connections: 20

Allow Root Login: No

You can also verify by looking at the config.xml

```
[root@SNgateway-v4_24_BETA-28 tmp]# grep -A 10 "sshserver subsystem"
/home/httpd/jbmconfig/conf/config.xml
    <sshserver subsystem="sshserver">
        <enable>y</enable>
        <advanced>y</advanced>
        <listenaddr>10.0.0.1</listenaddr>
        <listenport>22</listenport>
        <protocol>2</protocol>
        <gracetime>120</gracetime>
        <maxstartups>20</maxstartups>
        <rootlogin>n</rootlogin>
    </sshserver>
```

Step 6. Prepare install.sh

To make your setting take change when you install your sdk application, add the following lines to install.sh.

```
# make my custom changes to SSH settings
cmd /usr/bin/perl /tmp/mySSHsettings.pl

# clean up
cmd /bin/rm -f /tmp/mySSHsettings.pl
```

As mentioned in step 4, save your perl script into some directory in your sdk package you will know about and that will be extracted during installation. Recommended to save your perl script in /tmp/ directory.

Also it is recommended to clean up your perl script once it has been executed so that it would not be accidentally run again messing up your setting down the road. Clean up script is show in above example

Controlling Cellular Connection

cellmodemconnect.pl

To control cellular connection, a `cellmodemconnect.pl` script can be used. Following are the option `cellmodemconnect.pl` takes.

Options	Infromation
Usage	cellmodemconnect.pl <start | con | nocon | clear | stop | reset | status>
start	Issues a modem reset. 'stop' must be called first.
stop	Stop cell connection and polling programs (backends)
con	Signal backend proccesses to attempt a data connection, temporarily overriding config.
nocon	Signal backend proccesses to stop a data connection, temporarily overriding config.
clear	Clear both con/nocon triggers. Device should go back to config setting
status	Report current connection status
reset	Hard modem reset

Executing `cellmodemconnect.pl` without any options, will default to `Usage` with following output.

```
[root@SNgateway-v4_23_RC-99-22 ~]# cellmodemconnect.pl

 cellmodemconnect.pl v1.5

 Usage: cellmodemconnect.pl <start | con | nocon | clear | stop | reset |
status>

 start  - Issues a modem reset.  'stop' must be called first.

 stop  -  Stop cell connection and polling programs (backends)

 con   -  Signal backend proccesses to attempt a data connection,
temporarily
          overriding config.

 nocon  - Signal backend proccesses to stop a data connection, temporarily
          overriding config.

 clear - Clear both con/nocon triggers.  Device should go back to config
setting

 status - Report current connection status

 Note: 'start' or 'reset' may reboot unit depending on modem.
```

Example Output

Here are few examples of **`cellmodemconnect.pl`** status. Actual status message may differ on your device depending on which modem is installed and interface configurations.

Status of Cell Modem Up

```
[root@SNgateway-v4_23_RC-99-22 ~]# cellmodemconnect.pl status
Interface wwan0 is up
```

Status of Cell Modem Off/Down

No cellular interface available. No sim card

```
[root@SNgateway-v4_24_BETA-48 tmp]# cellmodemconnect.pl status

No Cell modem interfaces are up
Activation status : Searching
Data connection : Data Retry
```

Cellular modem is off. Sim card is available and active.

```
[root@SNgateway-v4_23_RC-99-22 ~]# cellmodemconnect.pl status
No Cell modem interfaces are up
Activation status : Reg Home
Data connection : Data Retry
```

Cellular statistics

Detailed cellular statistics are located in **`/var/log/wireless.cardstats`** log file. For more details, see System Statistics page.

- Cellular statistics in **`/var/log/wireless.cardstats`** will not be updated if cellular modem is off.
- **`/var/log/wireless.cardstats`** does not get removed when cell modem is stopped. It is left at last updated state.
- During cell modem start up / reset, **`/var/log/wireless.cardstats`** will be removed until modem is back up and running.

SMS Messages

Basic Instructions for Sending and Receiving of SMS Messeges with SN-SDK

This document pertains to Sixnet / Red Lion build versions 3.12 and higher.

Overview

Compatible Sierra Modules

Following modules are compatible for SMS messaging. SMS messaging may be limited by the plan provided by the cellular provider.

- MC572x (CDMA)
- MC7304, MC7330, MC7354
- MC7770
- MC8705, MC8790, MC8795

SMS Directory Structure

Directories should be created automatically for storing SMS files:

`/tmp/sms/send/` - Directory to hold SMS messages in queue until sent.

`/tmp/sms/send_fail/` - Directory to hold any SMS messages that have failed to be sent.

`/tmp/sms/send_ok/` - Directory to hold all SMS messages that have been sent successfully. (Available from version 3.18/4.18 and up).

`/tmp/sms/recv/` - Directory to hold SMS messages that the unit have recieved.

SMS Messeges in **`send_fail, send_ok`** and **`recv`** directories will be kept for 24 hours or until reboot.

SMS Character set

Only ASCII characters are accepted for SMS messages. Unicode characters could be sent/received, but will not be readable.

Sending SMS Messages

SMS data should be written to a plain text file with a unique file name, ex: /tmp/sms_send-01-02-03, and then moved/copied into /tmp/sms/send/.

Actual file name is irrelevant. Unique existence in the directory is important.

Format for SMS file for sending :

```
TO=6185551212
MSG=How are you?
```

- The **TO=** field is the recipient's phone number or text number. Must be at least 4 digits long. No dashes.
- The **MSG=** field is the message to send.
- The **MSG=** field must be at the bottom of the file so it can contain any <CR> and <LF> characters.
- Max characters is 240. Messages larger than 240 characters will be truncated.

Once this /tmp/sms_send-01-02-03 file is written, it must be moved to the **`/tmp/sms/send/`** directory. With this in mind, when a program is sending an SMS message, it should write out a temporary file with all of the data in it and then use the Linux mv -f command to move the file into place for completeness.

Example

1. Program writes file /tmp/sms_send-01-02-03 with content:

   ```
   TO=6185551212
   MSG=Test Message 123
   ```

2. Program runs the command

   ```
   mv -f /tmp/sms_send-01-02-03 /tmp/sms/send/
   ```

3. The file **`/tmp/sms/send/sms_send-01-02-03`** should disappear in a few seconds. If it does not, the system might be getting an error sending the message. After 3 attempts, the file will be moved to the **`/tmp/sms/send_fail/`** directory for 24 hours.
4. Multiple files may be placed into the send/ directory. They will be processed in order of timestamp, oldest first.

Output

All successfully sent messages are copied to the **`send_ok`** directory. A time stamp is appended to the file name with format `YYYYMMDD-hhmmss`.

All messages that failed to be sent, are copied to the **`send_fail`** directory. A time stamp is appended to the file name with format `YYYYMMDD-hhmmss.`

Success log output

In the syslog file, one should see something like this for GSM modems (**MC8790**, **MC8795**, **MC7700**, **MC8705**):

```
Oct 18 09:50:09 generic_watch: Reading SMS File
/tmp/sms/send/sms_send=01-02-03
Oct 18 09:50:10 generic_watch: Send Success: TO=6185551212, MSG=Test
Message 123
```

In the syslog file, one should see something like this for GSM modems (**MC7304**, **MC7330**, **MC7354**):

```
Mar 15 21:38:05 swi_qmi_watch[1577]: Reading SMS File
/tmp/sms/send/smsTest.txt
Mar 15 21:38:05 swi_qmi_watch[1577]: SMS Send Mode: CDMA/3GPP
Mar 15 21:38:05 swi_qmi_watch[1577]: Attempting to send SMS Message
TO=6185551212, MSG=Just a test, ID=3
Mar 15 21:38:06 swi_qmi_watch[1577]: Send Success: TO=6185551212, MSG=Just
a test
```

In the syslog file, one should see something like this for CDMA modems (**MC5727**, **MC5728**):

```
Oct 18 11:22:49 jbm_swi_vz: Reading SMS File
/tmp/sms/send/sms_send=01-02-03
Oct 18 11:22:50 jbm_swi_vz: SendSMS: Attempting to Send SMS Message
Oct 18 11:22:53 jbm_swi_vz: Notify: SMS message sent.
Oct 18 11:22:53 jbm_swi_vz: Notify: Call disconnected, Call State : 0x4040,
Reason : 10
```

The Call disconnected message does not mean the data connection dropped. This is the SMS call closing a channel.

Failed log output

A failure should look something like this for GSM modems (**MC8790**, **MC8795**, **MC7700**, **MC8705**):

```
Oct 18 09:51:17 generic_watch: Reading SMS File
/tmp/sms/send/sms_send=01-02-03Oct 18 09:51:17 generic_watch: Got ERROR
(not CMS error) trying to send SMS to 'fail'
Oct 18 09:51:17 generic_watch: Send #1 Failed: TO=6185551212, MSG=this
message should fail?? (truncated)
Oct 18 09:52:04 generic_watch: Reading SMS File
/tmp/sms/send/sms_send=01-02-03
Oct 18 09:52:04 generic_watch: Got ERROR (not CMS error) trying to send SMS
to 'fail'
Oct 18 09:52:04 generic_watch: Send #2 Failed: TO=6185551212, MSG=this
message should fail?? (truncated)
Oct 18 09:52:50 generic_watch: Reading SMS File
/tmp/sms/send/sms_send=01-02-03
Oct 18 09:52:50 generic_watch: Got ERROR (not CMS error) trying to send SMS
to 'fail'
Oct 18 09:52:50 generic_watch: Send #3 Failed : TO=6185551212, MSG=this
message should fail?? (truncated)
Oct 18 09:52:50 generic_watch: Max send attempt, moving file to
/tmp/sms/send_fail/sms_send=01-02-03.20121018-
```

A failure should look something like this for GSM modems (**MC7304**, **MC7330**, **MC7354**):

```
Mar 16 12:56:43 swi_qmi_watch[25164]: Reading SMS File
/tmp/sms/send/testsms
Mar 16 12:56:43 swi_qmi_watch[25164]: SMS Send Mode: UMTS/3GPP
Mar 16 12:56:43 swi_qmi_watch[25164]: Attempting to send SMS Message
TO=6185551212, MSG=Hi how are you, ID=4
Mar 16 12:56:43 swi_qmi_watch[25164]: ERROR: SLQSSendSMS() returned 0x41C
(eQCWWAN_ERR_QMI_DEVICE_NOT_READY), unable to send SMS, code 0xFFFFFFFF
Mar 16 12:56:43 swi_qmi_watch[25164]: Send #1 Failed: TO=6185551212, MSG=Hi
how are you (truncated)
Mar 16 12:56:54 swi_qmi_watch[25164]: Reading SMS File
/tmp/sms/send/testsms
Mar 16 12:56:54 swi_qmi_watch[25164]: SMS Send Mode: UMTS/3GPP
Mar 16 12:56:54 swi_qmi_watch[25164]: Attempting to send SMS Message
TO=6185551212, MSG=Hi how are you, ID=5
Mar 16 12:56:54 swi_qmi_watch[25164]: ERROR: SLQSSendSMS() returned 0x41C
(eQCWWAN_ERR_QMI_DEVICE_NOT_READY), unable to send SMS, code 0xFFFFFFFF
Mar 16 12:56:54 swi_qmi_watch[25164]: Send #2 Failed: TO=6185551212, MSG=Hi
how are you (truncated)
Mar 16 12:57:06 swi_qmi_watch[25164]: Reading SMS File
/tmp/sms/send/testsms
Mar 16 12:57:06 swi_qmi_watch[25164]: SMS Send Mode: UMTS/3GPP
Mar 16 12:57:06 swi_qmi_watch[25164]: Attempting to send SMS Message
TO=6185551212, MSG=Hi how are you, ID=6
Mar 16 12:57:06 swi_qmi_watch[25164]: ERROR: SLQSSendSMS() returned 0x41C
(eQCWWAN_ERR_QMI_DEVICE_NOT_READY), unable to send SMS, code 0xFFFFFFFF
Mar 16 12:57:06 swi_qmi_watch[25164]: Send #3 Failed : TO=6185551212,
MSG=Hi how are you
Mar 16 12:57:06 swi_qmi_watch[25164]: Max send attempt, moving file to
/tmp/sms/send_fail/testsms.20160316-125706
```

A failure should look something like this for CDMA modems (**MC5727**, **MC5728**):

```
Oct 18 11:26:50 jbm_swi_vz: Reading SMS File
/tmp/sms/send/sms_send=01-02-03
Oct 18 11:26:54 jbm_swi_vz: SendSMS: Attempting to Send SMS Message
Oct 18 11:26:56 jbm_swi_vz: Notify: Retry Sending SMS message later.
Oct 18 11:26:56 jbm_swi_vz: Notify: Call disconnected, Call State : 0x0,
Reason : 10
Oct 18 11:27:27 jbm_swi_vz: SendSMS: Attempting to Send SMS Message
Oct 18 11:27:30 jbm_swi_vz: Notify: Retry Sending SMS message later.
Oct 18 11:27:30 jbm_swi_vz: Notify: Call disconnected, Call State : 0x0,
Reason : 10
Oct 18 11:28:01 jbm_swi_vz: SendSMS: Attempting to Send SMS Message
Oct 18 11:28:03 jbm_swi_vz: Notify: Retry Sending SMS message later.
Oct 18 11:28:03 jbm_swi_vz: Notify: Call disconnected, Call State : 0x0,
Reason : 10
Oct 18 11:28:34 jbm_swi_vz: SendSMS: Attempting to Send SMS Message
Oct 18 11:28:36 jbm_swi_vz: Notify: Call disconnected, Call State : 0x0,
Reason : 10
Oct 18 11:28:38 jbm_swi_vz: SMS Send Failed : TO=6185581626, MSG=Test
Message 123 (truncated to 50 chars)
Oct 18 11:28:38 jbm_swi_vz: Moving SMS file to
/tmp/sms/send_fail/sms_send=01-02-03.20121018-112838
```

The Call disconnected message does not mean the data connection dropped. This is the SMS call closing a channel.

Receiving SMS Messages

SMS messages received by the modem will show up in the directory /tmp/sms/recv/. That directory may not appear until a message is actually received. SMS files in the /tmp/sms/recv/ will stay for 24 hours system time, or until the unit is rebooted.

Saved recieved SMS messages get the following file naming time stamp format:

sms_msg**YYYYMMDD**-**hhmmss**.**x**

Where

- **YYYY** = Year
- **MM** = Month
- **DD** = Day
- **hh** = Hour
- **mm** = Minute
- **ss** = Second
- **x** = an incremental digit that goes from 1 to 10 and then start over regardless of the time stamp.

File format of received SMS Message :

```
FROM=6185551212
TIME=10/18/12 - 08:49:26
PRI=Normal
MSG=Fine. How are you?
```

- The **TIME=** is a date time field when the message was received
- The **PRI=** is optional and may or may not be included.

- The **MSG=** will always be the last field and include all test until the end of the file, including <LF>/<CR>

Example Output

Below is a system log of SMS Message recieved

```
Mar 15 15:59:46 swi_qmi_watch[1400]: Received SMS Message Storage Type: 0,
Message Index: 0
Mar 15 15:59:47 swi_qmi_watch[1400]: Read UIM memory successful Message
Tag: 1, Message Format: 6
Mar 15 15:59:47 swi_qmi_watch[1400]: SMS RX Mode: UMTS
Mar 15 15:59:47 swi_qmi_watch[1400]: Got SMS from 9116185581212 at 15/03/16
- 15:59:45, Message: This is a test
```

Addendums

Here are few notes to keep in mind when working with SMS messages.

1. If a unit reboots or looses power, **all** (this includes received, sent okay, failed and to be sent) messages will be lost.
2. A unit will not send messages if the modem is not activated.
3. A unit will not send messages during an activation or a PRL update.
4. A unit will not send or receive messages if SMS is not included with the account. This may or may not report errors when sending.
5. Incoming SMS messages currently only are written to files with the date/timestamp in the file name in /tmp/sms/recv/, example : sms_msg20121018-130032
6. All SMS messages are deleted once they are 24 hours old (send_fail, send_ok and received).
7. Advanced SMS features such as attachments (pictures, MMS), and read/delivered receipts are not supported.

Sending Emails

Overview

Basic Instructions for Sending Email Messages with SN-SDK

This document pertains to Red Lion build versions 4.22 and higher.

Configuring the Device to Send Emails

In order to be able to send emails, the device needs to be configured. Once Email Client is configured and enabled, a **`/tmp/email/`** directory will be created.

If **`the /tmp/email/`** directory does not exist, then this means that email client is either disabled or not configured.

Configuring with GUI

To configure Email Client with GUI interface,

1. Go to **Services Email** from the device GUI interface
2. Select **Yes** from **Enable Email Support**
3. Enter your email server setting and username password.
4. Click **Apply**
5. From **Email Settings Test**, enter recipient email and click **Test Email** to verify that you have entered correct email settings.

For more information on Email Client, please see **Email Client** section from device User Guide.

Configuring with XFGlib

To configure with XFGlib, run the following script with the email server details

```
require "/etc/jbm/xfglib.pl";

my %xml_settings = (
            enable => "y",
   server => "smtp.youremailserver.com",
            port => "465",
   sender => "senderemail@youremailserver.com",
            username => "senderemailusername",
            password => "senderemailpassword"
            );
&xfg_set_multi_attribute("email", \%xml_settings);

&xfg_commit("apply");
```

For more information on XFGlib, please XFGlib documentation and XFGlib User Guide.

Email Message Format

Email Message File Format

The Email message file is a plain text file. All HTML tags will not be interpreted. The Email message **must** contain the following two lines: **`To:`** and **`Subject:`**

```
To: myemail@myserver.net
Subject: Email test
```

- The **To:** field is the recipient's email address. Must be in a valid email address format. If an invalid or non existing email is entered, but with a valid format, the email will be successfully sent. All failed to deliver notifications will be in the email box on the server. Failed to deliver notification cannot be detected by Ram Devices.
- The **Subject:** field is the subject line for the email. **`Subject:`** field may be left blank/empty, but characters "**`Subject:`**" **MUST** be present.

The **`To:`** and **`Subject`**: lines can be in either order, but must be the first two line of the message file. Everything else will be in the body of the email message.

Example of Email message file

```
To: myemail@myserver.net
Subject: Sample Email Test

1 Monday Tuesday Wednesday Thursday Friday Saturday Sunday Monday Tuesday
Wednesday Thursday Friday Saturday Sunday Monday Tuesday Wednesday Thursday
Friday Saturday Sunday Monday Tuesday Wednesday Thursday
2 January February March April May June July August September October
November December January February March April May June July August
September October November December January
```

Message body length

The outer limit of message body has not been tested and may depend on the email service provider.

A 2MB message has been successfully tested and sent with Gmail.

Email Character set

Email messages can contain any set of characters. The limitation may be based on your email server.

Sending Emails

For email messages to be sent, all email message files need to be copied and or moved to **`the /tmp/email`** directory. Once a message is in the directory, it will be sent automatically.

Example

1. Program writes the file **`/tmp/emailTest.txt`** with the content:

```
To: myemail@myserver.net
Subject: Sample Email Test
1 Monday Tuesday Wednesday Thursday Friday Saturday Sunday
```

2. Program runs the command

```
mv -f /tmp/emailTest.txt /tmp/email
```

3. The file **`/tmp/email/emailTest.txt`** should disappear in a few seconds.

Output

Success log output

In the syslog file, the following should be seen when email is sent successfully:

```
Mar 17 12:48:33 EmailAction: Attempting to send the email in emailTest.txt
file
Mar 17 12:48:34 EmailAction: successfully sent the email in emailTest.txt
file
```

Unsuccessful log output

In the syslog file, one may see something like this when email is was not sent successfully:

Failed to send Example 1

```
Mar 17 12:47:21 EmailAction: Rejected: The emailTest.txt file does not
appear to be a valid email, deleting the emailTest.txt file
```

Failed to send Example 2

```
Mar 17 12:23:54 EmailAction: Attempting to send the email in emailTest.txt
file
Mar 17 12:23:54 EmailAction: Could not send the email in emailTest.txt
file, reason UNKOWN
```

Sending Emails with Attachments

Sending emails with attachments is available in version 4.24 and later.

To send emails with attachments, run the following command **`email_attachment`**. Placing emails and attachments into the **`/tmp/email/`** directory **will not** send emails with attachments.

email_attachment

`/usr/local/bin/email_attachment` allows you to send email with attachments.

```
/usr/local/bin/email_attachment[-d] [-e] <PathToEmailMessage> [-a]
<PathToAttachment> <PathToAnotherAtachment> [-s] [-z]
```

Option	Requirement	Definition
-h <help>	-	Displays the usage information summary you are now reading
-d <debug mode>	Optional	print debug messages to system log.
-e <email file>	Required	Specify path to email message. Must be only one email message file.
-a <attachment(s)>	Optional	Specify path to attachment file. Can be multiple attachments.
-s <standalone>	Optional	Send the email as standalone via email_attachment script
-z <files.zip>	Optional	Zip the attachment(s) before sending the email. All attachments will be archived into **files.zip**

If option -a and attachments are not specified, the email will be sent without attachments.

Attachment Types

Attachments can be of any type and size. Total attachments file size should be no greater than 20MB.

Example

Let say we have an email we want to send with following attachments located in **`/vault/myprocessdata/`**

```
sampleLog.log
sampleZip.zip
sampleText.txt
ReadMeFile
emailMessage.txt
```

We would execute the following

```
email_attachment -e /vault/myprocessdata/emailMassage.txt -a
/vault/myprocessdata/sampleLog.log /vault/myprocessdata/sampleZip.zip
/vault/myprocessdata/sampleText.txt /vault/myprocessdata/ReadMeFile
```

or for simplicity

```
cd /vault/myprocessdata/
email_attachment -e emailMassage.txt -a sampleLog.log sampleZip.zip
sampleText.txt ReadMeFile
```

System Statistics

- System Information
    - General Device Information
        - DEVINFO_CONNECTION_STATUS
        - DEVINFO_TIME_SOURCE
        - DEVINFO_DEVICE_RESETTING
    - Cellular Statistics
    - Data Usage (vnstat)
    - On Board Statistics
    - Version

System Information

Several commands exist to gather specific information about the device that can be executed by command line. The examples below are used with a RAM 9931 unit without active cellular capabilities. Results of these commands may vary.

General Device Information

Executing "device_info.pl" will display general information about the device.

device_info.pl

```
DEVINFO_MANUFACTURER=Red Lion
DEVINFO_MODEL_NUMBER=RAM-9931
DEVINFO_MODEL_PLATFORM_TYPE=RAM
DEVINFO_MODEL_PLATFORM=9XXX
DEVINFO_MODEL_FAMILY=99XX
DEVINFO_DEVICE_SERIALNO=123X45678910112
DEVINFO_DEVICE_MODEM_ID=12345678
DEVINFO_DEVICE_PRL=
DEVINFO_DEVICE_RSSI=-125
DEVINFO_DEVICE_BARS=0
DEVINFO_DEVICE_MDN=Unavailable
DEVINFO_CONNECTION_STATUS=0
DEVINFO_TIME_SOURCE=0
DEVINFO_DEVICE_RESETTING=0
```

DEVINFO_CONNECTION_STATUS

This field checks if the flag file "wirelessdial_dialing" exists. This file represents a connection that was made or a connection that is being attempted. The value is either 0 for no connection, or 1 for connected.

DEVINFO_TIME_SOURCE

This field represents which time source the device is syncing to. The value is either 0 for no syncing, 1 for syncing to the cell modem, or 2 syncing to NTP time.

DEVINFO_DEVICE_RESETTING

This field us used to determine is something is wrong and the modem needs to reset. If the modem is resetting, the value is 1, otherwise it is 0.

Cellular Statistics

- Not all fields are available on all modules.
- /var/log/wireless.cardstats will not be updated if cellular modem is off.

Executing the "cat /var/log/wireless.cardstats" command will display cellular information about the device.

cat /var/log/wireless.cardstats

```
#File generated 2016-03-13 00:07:01

#Device Properties :
CELLMODEM_ESN=80FFFFFF
CELLMODEM_IMEI=111111111111111
CELLMODEM_MEID=11111111111111
CELLMODEM_SIM_STATUS=Not Present
CELLMODEM_SIM_ID=Unknown
CELLMODEM_SIM_IMSI=Unavailable
CELLMODEM_SIM_CARRIER_NUMBER=Unknown
CELLMODEM_SIM_CARRIER=Unknown
```

```
CELLMODEM_MDN=Unavailable
CELLMODEM_MIN=Unavailable
CELLMODEM_MODEL=MC7354
CELLMODEM_FIRMWARE_V=SWI9X15C_05.05.16.03 r22385 carmd-fwbuild1 2014/06/04
15:01:26
CELLMODEM_FIRMWARE_NUM=05.05.16.03
CELLMODEM_FIRMWARE_CARRIER=Verizon Wireless
CELLMODEM_FIRMWARE_PKGVER=014
CELLMODEM_FIRMWARE_PRIVER=005.013
CELLMODEM_HARDWARE_V=1.0
CELLMODEM_TEMPERATURE=34
CELLMODEM_TEMPERATURE_INFO=Normal
CELLMODEM_CONF_RADIO_TECH_PREF=
CELLMODEM_CONF_RADIO_TECH_PREF_E=0
CELLMODEM_CONF_RADIO_TECH_PREF_BITS=0
CELLMODEM_CURR_RADIO_TECH_PREF_BITS=31
CELLMODEM_CURR_RADIO_TECH_PREF_BITS_DESC=auto

#Network Service Info :
CELLMODEM_CARRIER=Unknown
CELLMODEM_CARRIER_PLMN=00
CELLMODEM_SERVICE_TYPE=No Service
CELLMODEM_SERVICE_TYPE_E=0
CELLMODEM_DATA_BEARER=Unknown
CELLMODEM_DATA_BEARER_E=-1
CELLMODEM_DATA_BEARER_CB=
CELLMODEM_DATA_BEARER_CB_E=0
CELLMODEM_RADIO_IF=CDMA 1xRTT
CELLMODEM_RADIO_IF_E=1
CELLMODEM_RADIO_IF_SSCB=
CELLMODEM_RADIO_IF_SSCB_E=0
CELLMODEM_RADIO_IF_RFCB=
CELLMODEM_RADIO_IF_RFCB_E=0
CELLMODEM_LTEINFO_BAND=47
CELLMODEM_LTEINFO_BANDWIDTH=Unknown
CELLMODEM_LTEINFO_RXCHAN=Unknown
CELLMODEM_LTEINFO_TXCHAN=Unknown
CELLMODEM_LTEINFO_EMM_STATE=Deregistered: No Cell
CELLMODEM_LTEINFO_EMM_CON_STATE=RRC Idle
CELLMODEM_OP_MODE=Online
CELLMODEM_SYS_MODE=LTE
CELLMODEM_SYS_MODE_E=9
CELLMODEM_IMS_REG_STATE=No Service
CELLMODEM_PS_STATE=Detached
CELLMODEM_BAND=Unknown
CELLMODEM_CHANNEL=0
CELLMODEM_CURRENT_RSSI=-125
CELLMODEM_CURRENT_ECIO=-125
CELLMODEM_CURRENT_IO=-125
CELLMODEM_CURRENT_SINR=-125
CELLMODEM_CURRENT_RSRQ=-125
CELLMODEM_CURRENT_RSRP=0
CELLMODEM_LTE_RSSI=-125
```

```
CELLMODEM_LTE_RSRQ=-125
CELLMODEM_LTE_RSRP=0
CELLMODEM_LTE_SINR=0
CELLMODEM_CDMA_RSSI=-125
CELLMODEM_CDMA_ECIO=0
CELLMODEM_HDR_RSSI=-125
CELLMODEM_HDR_ECIO=0
CELLMODEM_HDR_SINR=0
CELLMODEM_HDR_IO=0
CELLMODEM_GSM_RSSI=-125
CELLMODEM_WCDMA_RSSI=-125
CELLMODEM_WCDMA_ECIO=0
CELLMODEM_TDSCDMA_RSSI=-125
CELLMODEM_TDSCDMA_RSCP=-125
CELLMODEM_TDSCDMA_ECIO=0
CELLMODEM_TDSCDMA_SINR=0
CELLMODEM_QMI_STATE=Disconnected
CELLMODEM_SESSION_IPV4_STATE=Disconnected
CELLMODEM_SESSION_IPV6_STATE=Disconnected
CELLMODEM_LAST_SESSION_END_EREASON=0
CELLMODEM_LAST_SESSION_END_REASON=None
CELLMODEM_CALL_STATE=Unknown
CELLMODEM_QMI_CALL_TIME=0
CELLMODEM_CUR_CHANNEL_TX_RATE=0
CELLMODEM_CUR_CHANNEL_RX_RATE=0
CELLMODEM_MAX_CHANNEL_TX_RATE=0
CELLMODEM_MAX_CHANNEL_RX_RATE=0
CELLMODEM_CREG=Not Registered, Not Searching
CELLMODEM_CREG_ABR=Not Reg
CELLMODEM_ROAMING=1
CELLMODEM_ROAMING_TXT=Roaming
CELLMODEM_CURRENT_TX_BYTES=0
CELLMODEM_CURRENT_RX_BYTES=0
CELLMODEM_SIM_CONT_APN0=1:vzwims
CELLMODEM_SIM_CONT_APN1=2:vzwadmin
CELLMODEM_SIM_CONT_APN2=3:vzwinternet
CELLMODEM_SIM_CONT_APN3=4:vzwapp
CELLMODEM_SIM_CONT_APN4=5:vzw800

CELLMODEM_BASESTATION_LAT_RAW=0
CELLMODEM_BASESTATION_LAT=0:0:0
CELLMODEM_BASESTATION_LAT_DEC=0.000000
CELLMODEM_BASESTATION_LONG_RAW=0
CELLMODEM_BASESTATION_LONG=0:0:0
CELLMODEM_BASESTATION_LONG_DEC=0.000000
CELLMODEM_SERVSYS_DEFAULT_ROAMING_IND=0
CELLMODEM_SERVSYS_CS_ATTACH_STATE=0
CELLMODEM_SERVSYS_PS_ATTACH_STATE=0
CELLMODEM_SERVSYS_SEL_NET=0
CELLMODEM_SERVSYS_NUM_RADIO_INTS=0
CELLMODEM_SERVSYS_RADIO_INT=None
CELLMODEM_SERVSYS_DATA_CAPABILITIES=None
CELLMODEM_SERVSYS_MCC=0
```

```
CELLMODEM_SERVSYS_MNC=0
CELLMODEM_SERVSYS_NET_DESC=Unknown
CELLMODEM_SERVSYS_SYSTEM_ID=0
CELLMODEM_SERVSYS_NETWORK_ID=0
CELLMODEM_SERVSYS_BS_ID=0
CELLMODEM_SERVSYS_DAYLIGHT_SAV=0
CELLMODEM_SERVSYS_LEAP_SECONDS=0
CELLMODEM_SERVSYS_LOCALTIME_OFFSET=0
CELLMODEM_SERVSYS_CDMA_P_REV=0
CELLMODEM_SERVSYS_GPP_TIMEZONE=0
CELLMODEM_SERVSYS_GPP_NET_DSTA_ADJ=0
CELLMODEM_SERVSYS_LAC=0
CELLMODEM_SERVSYS_CELL_ID=0
CELLMODEM_SERVSYS_CONC_SVC_INFO=0
CELLMODEM_SERVSYS_CONC_PRL_IND=0
CELLMODEM_SERVSYS_CONC_DTM_IND=0
CELLMODEM_SERVSYS_SRV_STATUS=No Service
CELLMODEM_SERVSYS_SRV_CAPABILITY=No Service
CELLMODEM_SERVSYS_HDR_SRV_STATUS=No Service
CELLMODEM_SERVSYS_HDR_HYBRID=System is not hybrid
CELLMODEM_SERVSYS_IS_SYS_FORBIDDEN=System is not a forbidden system
CELLMODEM_SERVSYS_CDMA_SYSINFO_EXT_MCC=0
CELLMODEM_SERVSYS_CDMA_SYSINFO_EXT_IMSI_11_12=0
CELLMODEM_SERVSYS_HDR_PERSONALITY=0
CELLMODEM_SERVSYS_TRACK_AREA_CODE=0
CELLMODEM_SERVSYS_CS_BAR_STATUS=0
CELLMODEM_SERVSYS_PS_BAR_STATUS=0

#Connection Config Info :
CELLMODEM_CONF_DATA_DIRECTIVE=eCONF_DATA_CONNECT_DISABLED
CELLMODEM_CONF_DATA_DIRECTIVE_E=0
CELLMODEM_CONF_APN=

#SMS Info :
CELLMODEM_SMS_MODE=Invalid / Unsupported
```

```
CELLMODEM_SMS_SENT=0
CELLMODEM_SMS_SENT_FAIL=0
CELLMODEM_SMS_RECV=0
```

Data Usage (vnstat)

Using the "vnstat" command with no arguments will produce the following result.

vnstat (no arguments)

```
rx      /      tx      /     total    /   estimated
 can0: Not enough data available yet.
  (eth0):
       Mar '16    429.00 MiB  /  503.24 MiB  /  932.24 MiB  /    2.35 GiB
     yesterday    106.25 MiB  /  166.17 MiB  /  272.41 MiB
         today     25.53 MiB  /   33.28 MiB  /   58.81 MiB  /      59 MiB

 eth1: Not enough data available yet.
  (usb0):
       Mar '16     35.84 MiB  /    9.90 MiB  /   45.74 MiB  /  113.00 MiB
     yesterday       629 KiB  /      71 KiB  /     700 KiB
         today      1.03 MiB  /     112 KiB  /    1.14 MiB  /      --

 wlan0: Not enough data available yet.
 wwan0: Not enough data available yet.
 wwan1: Not enough data available yet.
 ip6tnl0: Not enough data available yet.
 dummy0: Not enough data available yet.
```

To get information about a particular interface, execute the command using the -i option followed by the name of the interface. Executing "vnstat -i eth0" will display information about eht0 exclusively.

vnstat -i eth0

```
Database updated: Sat Mar 12 23:20:22 2016

    (eth0) since 12/31/69

          rx:  646.54 MiB      tx:  727.70 MiB      total:  1.34 GiB

   monthly
                     rx      |     tx      |    total    |   avg. rate
     ------------------------+-------------+-------------+---------------
       Mar '16    429.10 MiB |  503.36 MiB |  932.46 MiB |    7.38 kbit/s
     ------------------------+-------------+-------------+---------------
     estimated      1.08 GiB |    1.27 GiB |    2.36 GiB |

   daily
                     rx      |     tx      |    total    |   avg. rate
     ------------------------+-------------+-------------+---------------
     yesterday    106.25 MiB |  166.17 MiB |  272.41 MiB |   25.83 kbit/s
         today     25.63 MiB |   33.39 MiB |   59.02 MiB |    5.75 kbit/s
     ------------------------+-------------+-------------+---------------
     estimated        25 MiB |      33 MiB |      58 MiB |
```

By executing "ls /storage/vnstat", a list of all the interfaces vnstat is keeping track of will be displayed.

ls /storage/vnstat

```
can0     dummy0   eth0     eth1     ip6tnl0  usb0     wlan0    wwan0
wwan1
```

While the output of vnstat is great for visualization, it is not practical to parse with a script. To get the information in a parse-able format, use the --exportdb option.

vnstat -i eth0 --exportdb

```
version;3
active;1
interface;eth0
nick;
created;0
updated;1457846818
totalrx;646
totaltx;727
currx;8445952
curtx;11831296
totalrxk;673
totaltxk;891
btime;1457817657
d;0;1457762408;25;33;775;572;0
d;1;1457676018;106;166;253;171;0
d;2;1457589601;44;19;38;331;0
```

```
d;3;1457503209;22;33;199;164;0
d;4;1457416825;22;32;783;79;0
d;5;1457330414;28;34;777;538;0
d;6;1457244022;32;35;222;909;0
d;7;1457157630;59;33;46;440;0
d;8;1457071220;30;33;660;485;0
d;9;1456984828;21;20;57;83;0
d;10;1456898427;19;33;507;278;0
d;11;1456812006;17;28;11;585;0
d;12;1456725630;11;16;305;236;0
d;13;1456639224;44;49;425;928;0
d;14;1456552825;48;47;270;423;0
d;15;1456466404;70;59;286;380;0
d;16;1456409188;1;1;555;520;0
d;17;1456207215;4;6;670;916;0
d;18;1456120821;5;0;401;287;0
d;19;1456051049;1;0;152;83;0
d;20;1455948013;6;8;300;249;0
d;21;1455879743;7;5;1021;955;0
d;22;1455775226;4;6;705;313;0
d;23;1455688802;5;11;712;890;0
d;24;1455614460;5;9;250;752;0
d;25;1455516028;0;0;533;588;0
d;26;1455429600;0;0;0;0;0
d;27;1455343399;0;0;0;0;0
d;28;1455172204;0;0;0;0;0
d;29;1455257176;0;0;0;0;0
m;0;1456812006;429;503;232;539;0
m;1;1454491943;217;224;441;352;0
m;2;1437197206;0;0;0;0;0
m;3;0;0;0;0;0;0
m;4;0;0;0;0;0;0
m;5;0;0;0;0;0;0
m;6;0;0;0;0;0;0
m;7;0;0;0;0;0;0
m;8;0;0;0;0;0;0
m;9;0;0;0;0;0;0
m;10;0;0;0;0;0;0
m;11;0;0;0;0;0;0
t;0;1457676018;106;166;253;171;0
t;1;1456466404;70;59;286;380;0
t;2;1456552825;48;47;270;423;0
t;3;1456639224;44;49;425;928;0
t;4;1457157630;59;33;46;440;0
t;5;1457244022;32;35;222;909;0
t;6;1457071220;30;33;660;485;0
t;7;1457589601;44;19;38;331;0
t;8;1457330414;28;34;777;538;0
t;9;1457503209;22;33;199;164;0
h;0;1457765982;1089;1465
h;1;1457769597;1995;1638
h;2;1457773186;3068;2475
h;3;1457776782;1140;1447
```

```
h;4;1457780394;824;1143
h;5;1457783999;966;1418
h;6;1457787574;1039;1407
h;7;1457791179;962;1417
h;8;1457794782;962;1418
h;9;1457798384;960;1417
h;10;1457801989;959;1418
h;11;1457805594;964;1418
h;12;1457809197;956;1418
h;13;1457812770;954;1405
h;14;1457816374;963;1418
h;15;1457819973;953;1364
h;16;1457823575;952;1418
h;17;1457827178;959;1418
h;18;1457830783;963;1420
h;19;1457834387;1005;1420
h;20;1457837988;1091;1416
```

```
h;21;1457841580;1123;1471
h;22;1457845171;994;1414
h;23;1457846818;534;701
```

On Board Statistics

To get information about on board statistics, the Read command can be executed. The Read command gathers information from IODB registers. Special reserved IODB registers contain system information. To access these special registers, the register type and the register address need to be known upon execution of the command.

> The Read command references addresses as 0 based.

Executing the Read command with no arguments will display a usage statement.

Read (No Arguments)

```
Usage: Read type startAddr numRegs
```

The following example executes the Read command with the type as 1 (Analog Out), the starting address as 1715 (RSSI Information), and the number of registers as 1 to get the information in the RSSI register.

Read 1 1715 1

```
65411
```

The full list of registers and their functionality are in the User Guide under Appendix B found in the device specific knowledge base link on the Documentation page. For more information about the Read command, follow this link.

Version

> Do Not Modify This File

By executing "cat /etc/version" the version information about the device will be displayed.

cat /etc/version

```
#Red Lion Version 4.24 160314 -- Tue Mar 15 00:21:11 CDT 2016
BUILD_MANU="Red Lion"
BUILD_VERSION="4.24.43.0"
BUILD_MAJORVER="4"
BUILD_MINORVER="24"
BUILD_MICROVER="43"
BUILD_RCVER="0"
BUILD_DATE="Tue Mar 15 00:21:11 CDT 2016"
BUILD_TARGET="btg25"
BUILD_ARCH="arm"
BUILD_TAG="160314"
BUILD_HASH="6430391df47961bfe2a82cc1b5352e1874781be8"
```

Getting GPS data from the Device

This page will talk about how to get GPS data from the device. There are four ways to get GPS data.

- gpsd port 2947
    - Changes to compile cpgs.c
        - In Makefile
        - In cgps.c
    - Example output
- NMEA Stream from /dev/gps_out
- GPS data parsing /var/log/wireless.gpscurrent
- GPS data from IODB registers

gpsd port 2947

Real time GPS data can be read directly from GPSd.

More information on gpds can be found here http://www.catb.org/gpsd/client-howto.html.

A good example of reading gps data with gpsd is in **`cgps.c`** from http://www.catb.org/gpsd/.

Changes to compile cpgs.c

Follwing changes are needed to me made to compile cgps.c:

In Makefile

In the make file, add the following libraries toe **`LDFLAGS`** list:

```
-lgps -lm -lncurses
```

In cgps.c

Since cgps.c is pure C code, but we use C++ complier, Extern "C" definition needs to be added to the file.

At the top of the cpgs.c file add the following

```
#ifdef __cplusplus
extern "C" {
#endif
```

and at the end of the file

```
#ifdef __cplusplus
}
#endif
```

Example output

Here is an example output from **`cgps.c`**

```
+-------------------------------------------++-----------------------------
-----+
|    Time:       2016-03-24T17:20:13.000Z   ||PRN:   Elev:  Azim:  SNR:
Used: |
|    Latitude:    34.727701 N               ||  37    00    000    00
Y   |
|    Longitude:   87.336630 W               ||  37    00    000    00
Y   |
|    Altitude:   724.4 ft                   ||  37    00    000    00
Y   |
|    Speed:      0.0 mph                    ||
|
|    Heading:    0.0 deg (true)             ||
|
|    Climb:      n/a                        ||
|
|    Status:     3D FIX (196 secs)          ||
|
|    Longitude Err:   n/a                   ||
|
|    Latitude Err:    n/a                   ||
|
|    Altitude Err:    +/- 105 ft            ||
|
|    Course Err:      n/a                   ||
|
|    Speed Err:       n/a                   ||
|
|    Time offset:     -214601.397           ||
|
|    Grid Square:     EM48tr                ||
|
+-------------------------------------------++-----------------------------
-----+
```

NMEA Stream from /dev/gps_out

If your application uses a NMEA parser, you could just tap into `/dev/gps_out` stream and parse for any GPS data needed.

GPS data parsing /var/log/wireless.gpscurrent

GPS data can be also gathered from parsing `/var/log/wireless.gpscurrent` log file.

Please note, `wireless.gpscurrent` log file is updated about every 5 seconds.

cat /var/log/wireless.gpscurrent

```
#===========
#GPS CONFIG
#===========
Version=2.04
ConfigFilename=/etc/jbm/gps/gpsc.conf
MaxGPSLogSize=600
IntervalToSaveGPSLocation=10
LockdownRadiusMultiplier=2.000000
TotalEntriesToSave=500
RequiredNumbOfValidEntries=200
RequiredNumbOfSatForValid=2
SecondsOutsideFenceB4Violation=60
SecondsAllowedOfNoDataB4Violation=120
NumbViolationPtsToIgnore=34
NumbOutlierPointsToDiscard=0
RequireUserToClearViolation=N
BootupLockdownUntilGPSSeen=N [N/A]
ViolationActionType=REPORTONLY
MeasurementStandard=FEET
DistanceReportingThreshold=1500
MinimumAccuracyFT=200
FixedLockdownRadius=50
UseSerialPort=d
SerialPortSpeed=0
USER_FIXED_LAT=0.000000
USER_FIXED_LONG=0.000000
GPSSourceNum=1
GPSSource=Internal
SecondsSinceLastGpsFix=1
#===========
#LOCKDOWN INFO
#===========
LOCKDOWN_ENGINESTATE=MONITORONLY
LOCKDOWN_ENGINESTATE_NUM=0
LOCKDOWN_CP_GPS_LAT=44.727609659
LOCKDOWN_CP_GPS_LONG=-94.337211733
LOCKDOWN_CP_GPS_ALT=187.4
LOCKDOWN_CP_GPS_TIMESTAMP=151214
LOCKDOWN_GPS_NUMSAT=22
LOCKDOWN_MIN_ACCURACY_CALC=129
LOCKDOWN_RADIUS_CALC=258
LOCKDOWN_RADIUS_CURRENT=0
#===========
#RADIUS INFO
#===========
#CFG: Lowest threshold allowed, from a User Fixed Entry or Engine Calc=200
#CFG: FixedLockdownRadius=50
#CFG: FixedRadius is in ABSOLUTE mode.
#Engine: Calculated Minimum Accuracy of Measurement=129
#Engine: Calculated Radius with Accuracy * Multiplier=258
#Engine: Current Radius in use=0
```

```
#Engine: Effective Range of movement before Violation=-129
#===========
#CURRENT INFO
#===========
CURRENT_GPS_VALID=1
CURRENT_GPS_LAT=44.727487117
CURRENT_GPS_LONG=-94.337204867
CURRENT_GPS_ALT=187.4
CURRENT_GPS_TIMESTAMP=151214
CURRENT_GPS_NUMSAT=23
CURRENT_GPS_FTFROMCP=45
CURRENT_GPS_SPEED=0
CURRENT_GPS_COURSE=257
CURRENT_GPS_HEADING=0
#===========
#Last Known Good Info
#===========
LAST_GPS_VALID=1
LAST_GPS_LAT=44.727487117
LAST_GPS_LONG=-94.337204867
LAST_GPS_ALT=187.4
LAST_GPS_TIMESTAMP=151214
LAST_GPS_NUMSAT=23
LAST_GPS_FTFROMCP=45
LAST_GPS_SPEED=0
```

```
LAST_GPS_COURSE=257
LAST_GPS_HEADING=0
#===========
```

GPS data from IODB registers

GPS IODB tag information can be found device GUI interface under **Automation > Tags**. GPS IODB registers are 1201 to 1222.

> Please note, GPS IODB values are updated about every 5 seconds.

Here is the example below to get IODB value with C code.

> Please note that IODB register mentioned on **Automation > Tags** are one (1) based, but IODB registers used with code are zero (0) based.

GPS IODB read with C code

```
int i;
short gps_values[22];

IODBRead(1, 1200, 22, (void *)&gps_values, NULL);

for (i = 0; i < 22; i++)
{
    printf("gps_values[%d]: %d\n", i, gps_values[i]);
}

// The output will be:
//gps_values[0]: 1606
//gps_values[1]: 5
//gps_values[2]: 1
//gps_values[3]: 38
//gps_values[4]: 43
//gps_values[5]: 40
//gps_values[6]: 0
//gps_values[7]: 44
//gps_values[8]: 7278
//gps_values[9]: 94
//gps_values[10]: 20
//gps_values[11]: 14
//gps_values[12]: 1
//gps_values[13]: -90
//gps_values[14]: 3372
//gps_values[15]: 20
//gps_values[16]: 2314
//gps_values[17]: 0
//gps_values[18]: 0
//gps_values[19]: 0
//gps_values[20]: 0
//gps_values[21]: 1
```

GPS IODB data can be also retrieved with Read command.

Read GPS IODB command

```
[root@SNgateway-v4_24_BETA-47 tmp]# Read 1 1200 22
      1606           5           1          44          43          40
0          38
      7278          94          20          14           1       65446
3372          20
      2314           0           0           0           0           1
```

AT Command Interface

- Input
- Output

    - Formatted (at_cmd_out)
    - Raw (at_cmd_out_raw)
- Sample Script

First available in 4.21

Input

Not available on 3G Sprint/Verison products like 66xx series units.

AT commands are executed by writing them to a file called "at_cmd_in" located in the /tmp directory.

Example of Running ATI command from command line.

```
echo "ATI" > /tmp/at_cmd_in
```

Output

While the script in the sample script section prints out the contents of the formatted file as seen below, the raw contents can be viewed by running the command "cat /tmp/at_cmd_out_raw".

Formatted (at_cmd_out)

Formatted Output

```
./test_at_cmd_in.sh
=================================================
 Sending : ATI
 Output :

ATI
Manufacturer: Sierra Wireless, Incorporated
Model: MC7354
Revision: SWI9X15C_05.05.16.03 r22385 carmd-fwbuild1 2014/06/04 15:01:26
MEID: 11111111111111
ESN: 1111111111, 80FFFFFF
IMEI: 111111111111111
IMEI SV: 13
FSN: J1111111111111
+GCAP: +CIS707-A, CIS-856, CIS-856-A, +CGSM, +CLTE2, +MS, +ES, +DS, +FCLASS


OK
=================================================
```

Raw (at_cmd_out_raw)

cat /tmp/at_cmd_out_raw

```
cat /tmp/at_cmd_out_raw
ATI
Manufacturer: Sierra Wireless, Incorporated
Model: MC7354
Revision: SWI9X15C_05.05.16.03 r22385 carmd-fwbuild1 2014/06/04 15:01:26
MEID: 11111111111111
ESN: 1111111111, 80FFFFFF
IMEI: 111111111111111
IMEI SV: 13
FSN: J1111111111111
+GCAP: +CIS707-A, CIS-856, CIS-856-A, +CGSM, +CLTE2, +MS, +ES, +DS, +FCLASS


OK
```

Sample Script

This script follows a simple procedure of writing to an a file called "at_cmd_in", followed by waiting for an "at_cmd_out" file, and then reading that file.

test_at_cmd_in.sh

```
#!/bin/bash

INFILE="/tmp/at_cmd_in"
OUTFILE="/tmp/at_cmd_out"
OUTFILERAW="/tmp/at_cmd_out_raw"

rm -f $OUTFILE
rm -f $OUTFILERAW

echo "ATI" > $INFILE
sleep 2
count=0
while [ $count -le 5 ]; do

 if [ -e "$OUTFILE" ]; then
  break
 fi
 sleep 1
 count=$(($count+1))
done

if [ -e "$OUTFILE" ]; then
 cat $OUTFILE
 exit
else
 echo "Error: File not present"
fi
```

Include Libraries

When writing your application, please use these libraries instead of supplying your own to avoid redundancy and duplicate libraries being installed on the device. If you need a library that is not already installed on the device, you may included it with your package. There are a few libraries available with EDLK toolchain that are not used by RAM devices so they are not installed onto the device.

> The device has a limited amount of space. Using static library includes and and supplying duplicate copies of shared libraries may result in application not being able to install or getting corrupted on re-flash.

Following libraries **are included on the RAM device** as of version 4.23.

- \lib\
- \usr\lib\

\lib

```
ld-2.6.so
ld-linux.so.3 -> ld-2.6.so
libBrokenLocale-2.6.so
libBrokenLocale.so.1 -> libBrokenLocale-2.6.so
```

```
libSegFault.so
libanl-2.6.so
libanl.so.1 -> libanl-2.6.so
libbz2.so -> libbz2.so.1.0.4
libbz2.so.1 -> libbz2.so.1.0.4
libbz2.so.1.0.4
libc-2.6.so
libc.so.6 -> libc-2.6.so
libcap.so.1 -> libcap.so.1.10
libcap.so.1.10
libconfuse.so -> libconfuse.so.0.0.0
libconfuse.so.0 -> libconfuse.so.0.0.0
libconfuse.so.0.0.0
libcrypt-2.6.so
libcrypt.so.1 -> libcrypt-2.6.so
libcrypto.so.0.9.8 -> libcrypto.so.1.0.0
libcrypto.so.1.0.0
libcrypto.so.2 -> libcrypto.so.1.0.0
libcurl.so -> libcurl.so.4.3.0
libcurl.so.4 -> libcurl.so.4.3.0
libcurl.so.4.3.0
libdl-2.6.so
libdl.so.2 -> libdl-2.6.so
libexslt.so -> libexslt.so.0.8.13
libexslt.so.0 -> libexslt.so.0.8.13
libexslt.so.0.8.13
libftdi1.so
libgcc_s.so.1
libgmp.so -> libgmp.so.3.4.4
libgmp.so.3 -> libgmp.so.3.4.4
libgmp.so.3.4.4
libgps.so.22 -> libgps.so.22.0.0
libgps.so.22.0.0
libgpsd.so.22 -> libgpsd.so.22.0.0
libgpsd.so.22.0.0
libiw.so -> libiw.so.30
libiw.so.30
liblzo2.so.2
libm-2.6.so
libm.so.6 -> libm-2.6.so
libmemusage.so
libnetfilter_conntrack.so -> libnetfilter_conntrack.so.3.0.1
libnetfilter_conntrack.so.3 -> libnetfilter_conntrack.so.3.0.1
libnetfilter_conntrack.so.3.0.1
libnfnetlink.so -> libnfnetlink.so.0.2.0
libnfnetlink.so.0 -> libnfnetlink.so.0.2.0
libnfnetlink.so.0.2.0
libnl.so -> libnl.so.1.1.4
libnl.so.1 -> libnl.so.1.1.4
libnl.so.1.1.4
libnsl-2.6.so
libnsl.so.1 -> libnsl-2.6.so
libnss_compat-2.6.so
```

```
libnss_compat.so.2 -> libnss_compat-2.6.so
libnss_dns-2.6.so
libnss_dns.so.2 -> libnss_dns-2.6.so
libnss_files-2.6.so
libnss_files.so.2 -> libnss_files-2.6.so
libnss_hesiod-2.6.so
libnss_hesiod.so.2 -> libnss_hesiod-2.6.so
libnss_nis-2.6.so
libnss_nis.so.2 -> libnss_nis-2.6.so
libnss_nisplus-2.6.so
libnss_nisplus.so.2 -> libnss_nisplus-2.6.so
libospf.so -> libospf.so.0.0.0
libospf.so.0 -> libospf.so.0.0.0
libospf.so.0.0.0
libospfapiclient.so -> libospfapiclient.so.0.0.0
libospfapiclient.so.0 -> libospfapiclient.so.0.0.0
libospfapiclient.so.0.0.0
libpam.so.0 -> libpam.so.0.81.6
libpam.so.0.81.6
libpcap.so -> libpcap.so.1.1.1
libpcap.so.1 -> libpcap.so.1.1.1
libpcap.so.1.1.1
libpcprofile.so
libpopt.so -> libpopt.so.0.0.0
libpopt.so.0 -> libpopt.so.0.0.0
libpopt.so.0.0.0
libproc-3.2.7.so
libpthread-2.6.so
libpthread.so.0 -> libpthread-2.6.so
libresolv-2.6.so
libresolv.so.2 -> libresolv-2.6.so
librt-2.6.so
librt.so.1 -> librt-2.6.so
libssl.so.0.9.8 -> libssl.so.1.0.0
libssl.so.1.0.0
libssl.so.2 -> libssl.so.1.0.0
libssp.so.0 -> libssp.so.0.0.0
libssp.so.0.0.0
libstdc++.so.6 -> libstdc++.so.6.0.9
libstdc++.so.6.0.9
libstunnel.so
libtermcap.so.2 -> libtermcap.so.2.0.8
libtermcap.so.2.0.8
libthread_db-1.0.so
libthread_db.so.1 -> libthread_db-1.0.so
libtinyxml.so -> libtinyxml.so.1.0.1
libtinyxml.so.1 -> libtinyxml.so.1.0.1
libtinyxml.so.1.0.1
libusb-0.1.so -> libusb-0.1.so.4.4.4
libusb-0.1.so.4 -> libusb-0.1.so.4.4.4
libusb-0.1.so.4.4.4
libusb-1.0.so -> libusb-1.0.so.0.0.0
libusb-1.0.so.0 -> libusb-1.0.so.0.0.0
```

```
libusb-1.0.so.0.0.0
libutil-2.6.so
libutil.so.1 -> libutil-2.6.so
libuv.so -> libuv.so.1
libuv.so.1
libvolume_id.so -> libvolume_id.so.1.0.9
libvolume_id.so.1 -> libvolume_id.so.1.0.9
libvolume_id.so.1.0.9
libwrap.so.0 -> libwrap.so.0.7.6
libwrap.so.0.7.6
libxml2.so -> libxml2.so.2.7.7
libxml2.so.2 -> libxml2.so.2.7.7
libxml2.so.2.7.7
libxslt.so -> libxslt.so.1.1.22
libxslt.so.1 -> libxslt.so.1.1.22
libxslt.so.1.1.22
libxtables.la
libxtables.so -> libxtables.so.0.0.0
libxtables.so.0 -> libxtables.so.0.0.0
libxtables.so.0.0.0
libz.so.1 -> libz.so.1.2.3
libz.so.1.2.3
```

```
libzebra.so -> libzebra.so.0.0.0
libzebra.so.0 -> libzebra.so.0.0.0
libzebra.so.0.0.0
```

\usr\lib\

```
libcli.so -> libcli.so.1.9.5
libcli.so.1 -> libcli.so.1.9.5
libcli.so.1.9 -> libcli.so.1.9.5
libcli.so.1.9.5
libcommonc++.so -> libcommonc++.so.1
libcommonc++.so.1
libddioc++.so -> libddioc++.so.1.0.2
libddioc++.so.1 -> libddioc++.so.1.0.2
libddioc++.so.1.0.2
libfileimonitorc++.so -> libfileimonitorc++.so.1
libfileimonitorc++.so.1
libfilevecc++.so -> libfilevecc++.so.1
libfilevecc++.so.1
libloggerc++.so -> libloggerc++.so.1
libloggerc++.so.1
libsxiodbc++.so -> libsxiodbc++.so.1.3.2
libsxiodbc++.so.1 -> libsxiodbc++.so.1.3.2
libsxiodbc++.so.1.3.2
libtimerc++.so -> libtimerc++.so.1
libtimerc++.so.1
libtokenstrtlutablec++.so -> libtokenstrtlutablec++.so.1
libtokenstrtlutablec++.so.1
```

snupdate

Building Packages for Sixnet / Red Lion Wireless Devices (D-Series; R3000; SN/RAM-6000; RAM-9000)

Rules for Packages

- All package names must end in .zip
- snupdate will only work if the package file in the /tmp directory
- snupdate does not support password protection on user-created packages
- It is recommended that users create packages on a unit which makes it easier to ensure that files have the correct paths, permissions, etc.

Modes of snupdate operation

snupdate has three modes of operation, based on the contents of the zip file it is called upon.

Unscripted Installation Packages

This is the default snupdate behavior. Unless it detects special contents in the zip file, snupdate treats the package as a plain zip file and extracts the contents with their full paths to the root directory of the device, thus all files should have the full path name when the package is created. This default behavior provides a simple way to update devices and install new files and programs, but does not provide the flexibility of the scripted installation method.

Scripted Installation Packages

If snupdate finds a file named install.sh in the root directory of a package, snupdate will treat it as a scripted installation package. This means that snupdate will only extract the install.sh file to tmp and then run /tmp/install.sh, passing it a single argument of the package file name. The install.sh script is expected to extract the rest of the contents of the package itself and perform any other operations that the package requires. This method allows maximum flexibility but adds some complexity that may not be necessary for simple packages. For example, a scripted installation would be useful for a package that needs to check the current status or configuration of a unit (e.g. the firmware version) and behave differently based on that information.

Sample install.sh

```
#!/bin/sh
# Check that /etc/version exists. This pretty much only true on JBM/SN
devices
if [ ! -e "/etc/version" ]; then
 echo "Unit does not appear to be a JBM/Sixnet unit, exiting."
 exit 1
fi

# The only argument we are passed is the file name of the package
FILE_NAME=$1

# Make sure we can find that file before we continue
if [ ! -e "$FILE_NAME" ]; then
 echo "Cannot find package file $FILE_NAME"
 exit 2
fi
# Extract the rest of the contents of the package
# -d The destination directory for the unpacked contents
# -o Overwrite existing files without prompting (use this!)
# $FILE_NAME The file name of the package to unzip
# -x install.sh Exclude install.sh when extracting since that has already
been unpacked
unzip -d / -o $FILE_NAME -x install.sh

# Now that the contents are extracted, we can do any other action the
package needs to perform perl /tmp/xmllib.gmuclient.pl
```

Configuration Only Packages (gatherconfigs)

If snupdate does not find an install.sh script, but it does find all three of: /home/httpd/jbmconfig/conf/config.xml, /etc/rc.d/rc.local, and /etc/hosts, it will treat the package as a configuration package created by a gatherconfigs. This provides a fairly simple method to save the configuration of the unit and to restore that config or transfer it to another unit.

Examples of Common Tasks in Scripted Installs

Check Firmware Version

```
# bring main version variables into local scope
source /etc/version

if [ "$BUILD_MAJORVER" -ne 4 ]; then
 echo "Package only applies to 4-series devices"
 exit 1
fi
if [ "$BUILD_MINORVER" -lt 22 ]; then
 echo "Package relies on library updates introduced in 4.22"
 exit 1
fi

echo "
Installed on:
Full Version $BUILD_VERSION
RC Version $BUILD_RCVER
"
```

Package Preservation

Storage Spaces

During the installation process, if the .zip file needs to be stored in a location that will persist after re-flash, then it needs to go into one of two places; Storage or Vault. Storage is a smaller partition that was used in older models when the Vault partition did not exist. It's purpose was to keep a section of memory that would maintain its data across re-flashes. As newer models with more space were introduced, a new version 4.23 was introduced containing a new partition with more memory available. The table below shows the size comparison between the Storage and Vault partitions.

Partition	Size	Version Supported
Storage	4 megabytes, 2 MB customer usable	4.22 and below
Vault	16 megabytes	4.23+

Partitions

By executing the "df -h" command, it produces the following result.

Partitions

```
Filesystem  Size Used Available Use%   Mounted on
/dev/root  120.0M  31.4M  88.6M  26%   /
/dev/mtdblock3  4.0M   1.8M   2.2M   44%   /storage
/dev/mtdblock4  3.0M   2.7M   284.0K 91%   /boot
tmpfsvar   5.0M   376.0K   4.6M   7%   /var
tmpfstmp   80.0M   72.0K   79.9M  0%   /tmp
tmpfsvar   32.0K   0    32.0K  0%   /media
/dev/mtdblock6  16.0M   692.0K   15.3M  4%   /vault
/dev/mtdblock9  253.0M   72.6M   180.4M 29%   /images
/dev/mtdblock10 32.0M   1.0M   31.0M  3%   /datalog
```

There are six major partitions that exist on the device. Four of them will have their data persist after re-flash. When installing a package, it is important to know where to store data that should not be changed, even after a firmware update.

Partition	Type	Persists After Re-flash	Persists after Advanced Re-flash	Persists After Reboot	Purpose
tmp	memory	No	No	No	Temporary Storage
var	memory	No	No	No	Program data during execution
storage	mtdblock	**Yes**	**No**	**Yes**	Re-flashing tools and saved configs
boot	mtdblock	No	No	Yes	Kernel
root	memory	No	No	Yes	Main System information
vault	mtdblock	**Yes**	**Yes**	**Yes**	Open
images	mtdblock	Yes	Yes	Yes	Wireless Module Firmware Storage
datalog	mtdblock	Yes	Yes	Yes	Data Logger records

Based on the table above, the Vault partition is the best place to preserve a package. It's data will be preserved across both re-flash and advanced re-flash.

Temporary Storage Warning
Be careful when installing parts of the program into the tmp directory as it is removed during re-flash, advanced re-flash, and system shut down.

Package Preserve Automation

Package Persevere Automation will be available in version 4.24 and later

To ensure that a install package is preserved across re-flashes, the **`system_package_preserve`** script was created to automate the copying of the install package to the correct partition. It will first try to copy it to the Vault partition and it one does not exist (it is an older model), then it will copy the package to the Storage partition.

Usage

The system_package_preserve requires only one argument. That argument is the name of the package zip file. The format for the script is as follows.

```
system_package_preserve "<package_name.zip>"
```

An example of using this script for an install package called "install.zip" would be to include the line below in the install.sh script for the package.

An example how it would look in your install.sh

```
system_package_preserve "install.zip"
```

Running from install.sh, your installer package is the first argument. Run system_package_preserve "$1" to have your application be reinstalled after a reflash.

GAU Custom Extensions

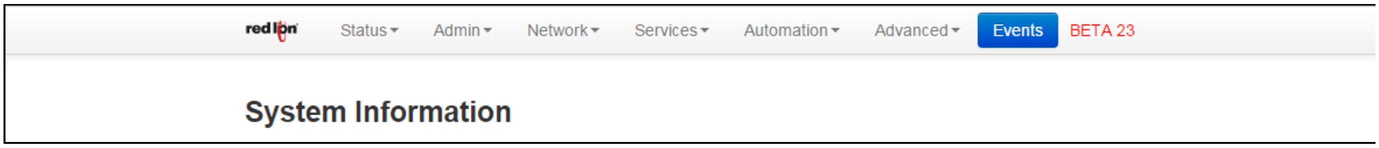
- General Information
- Usage
- SDK Installation Implementation
- Notes
- Custom tabs on version 4.23.

GAU Custom Extensions will be available in version 4.24 and later.

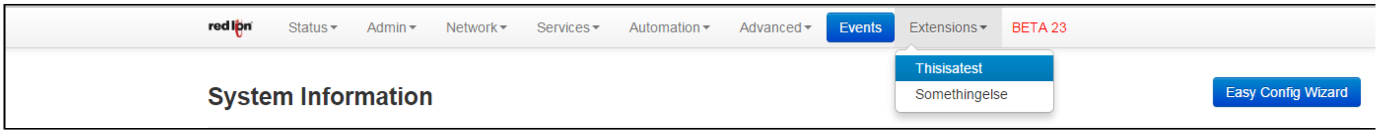
General Information

The GAU interface supports the ability to add a link from the main GAU page to custom made pages through the Extensions tab.

If no custom made pages have been added to the customTabs.txt file using the **webui_extension_add** command, then the GAU interface will look like the image below.

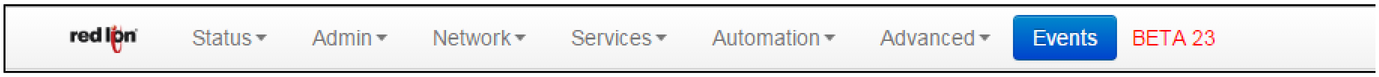


After utilizing the **webui_extension_add** command, an "Extensions" tab will be displayed with a drop down list containing all the custom pages as shown below.

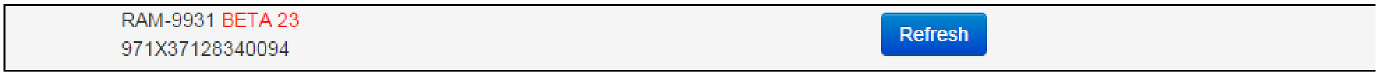


All of the HTML and Javascript documents for the GAU interface are stored in the **/home/httpd/jbmconfig/html/pages** directory. The GAU interface uses Pager.js to take a list of webpage URLs and attach the GAU header and footer to the webpage.

Header



Footer



By using the **webui_extension_add** command, a custom made page entry can be added to the customTabs.txt file. However, if the HTML and Javascript files are not in the pages directory but were added to the customTabs.txt using the webui_extension_add command, the custom page will not be displayed.

Usage

Usage Format: webui_extension_add [-h | -r | -p | -u | -m] "<Title>" "<URL>"

-h: Displays the usage text and then exits.

Ex. webui_extension_add -h

-r: The <URL> argument will be written to the configuration file without the "#!/" characters. This option should be used if the custom page will not have the GAU header or footer.

Ex. webui_extension_add -r "Start Page" "start"

Start Page,start

-p: The <URL> argument will be written to the configuration file with the "#!/" characters. This option should be used if the custom page will have the GAU header and footer.

Ex. webui_extension_add -p "Start Page" "start"

Start Page,#!/start

-u: Deletes the entry in the configuration file containing the <Title> argument.

Ex: webui_extension_add -p "Start Page"

"Start Page,#!/start" will be deleted.

-m: Displays the extended usage statement.

Note: The <Title> and <URL> arguments must be inside of quotes. If no quotes are used, each word spaced out will become it's own argument and will cause a usage error.

SDK Installation Implementation

The GAU custom extension(s) can be added to the installation process by adding the command(s) to the install.sh file. Using the code from the sn update page, the webui_extension_add command can be entered after the unzip command as shown in the image below.

Install.sh

```
#!/bin/sh
# Check that /etc/version exists. This pretty much only true on JBM/SN
devices
if [ ! -e "/etc/version" ]; then
 echo "Unit does not appear to be a JBM/Sixnet unit, exiting."
 exit 1
fi

# The only argument we are passed is the file name of the package
FILE_NAME=$1

# Make sure we can find that file before we continue
if [ ! -e "$FILE_NAME" ]; then
 echo "Cannot find package file $FILE_NAME"
 exit 2
fi
# Extract the rest of the contents of the package
# -d The destination directory for the unpacked contents
# -o Overwrite existing files without prompting (use this!)
# $FILE_NAME The file name of the package to unzip
# -x install.sh Exclude install.sh when extracting since that has already
been unpacked
unzip -d / -o $FILE_NAME -x install.sh

# NEW ADDITION
# Now that the contents are extracted, we can do any other action the
package needs to perform perl /tmp/xmllib.gmuclient.pl
# If a custom webpage is to be added to the extensions tab on the GAU
website
# then use the webui_extension_add command.
# -p The webpage will use the GAU header and footer and will have "#!/"
# prepended to it in the customTabs.txt file.
# -r The webpage will not use the GAU header and footer
# -h This will display the usage message
# The command below will add an item called "Page Title" under the
extensions tab
# and will link to the page "my_page". This page will use the GAU header
and footer.
webui_extension_add -p "Page Title" "my_page"
```

Notes

All custom pages listed in the Extensions tab must have an appropriate Javascript file. The bare minimum would look like code below.

myPage.js

```
define ([], function() {

// Custom Code Here

});
```

While the code above would allow the custom page would be displayed, the format and style will not match the GAU. To get the full compatibility, the Javascript file should look more like the code below.

myBetterPage.js

```
define (["jquery", "knockout"], function ($, ko){

// Custom Code Here

});
```

Custom tabs on version 4.23.

If you have firmware version 4.23 or would like to manually add a custom tab without **webui_extension_add** command described above, please see Add Custom Tabs to Navigation page.

Add Custom Tabs to Navigation

> **Please Note**
> This page is for adding custom tabs to navigation on firmware 4.23. If you have version 4.24 and above, you may also use this method, if desired, but it is recommended to use **webui_extension_add** command described on GAU Custom Extensions page.

Custom tabs are defined by a csv file on-device. With this file populated, entries will show up in order under the tab **Custom**, following **Advanced** on the main navigation bar.

Update via the GUI

1. Navigate to **Advanced** > **Expert Mode** > **Configure Sub-systems** through the GUI
2. Select **Custom Tabs** from the drop-down
3. Add a line to the csv file similarly to the commented examples
    - Use a pound-sign (#) at the beginning of a line to remove (comment-out) an entry
4. Click **Save**
5. Observe that your tab appears under the new entry **Custom** on the navigation bar

Update via Script

You can modify the custom tabs configuration file with a script. The following snippet will add or update an entry in the main configuration file

```
PAGE_TITLE="My Custom Tab"            # This will be the text of the tab

PAGE_URL="mypage"                     # If your custom page contains an
index.html
PAGE_URL="mypage/not_index.html"      # If your custom page does not
contain an index.html
PAGE_URL='#!/start'                   # Link to existing page, note use of
single-quotes (! is a special character)
PAGE_URL="http://www.redlion.net/"    # External URL

TABSFILE="/home/httpd/jbmconfig/txt/customTabs.txt"

# See if the entry already exists
if ! grep "^${PAGE_TITLE}," $TABSFILE &>/dev/null; then
 # If not, append it to the list
 echo "${PAGE_TITLE}, ${PAGE_URL}" >> $TABSFILE
else
 # Otherwise, update it
 PAGE_URL="${PAGE_URL//\//\\/}" # escape any forward-slashes for the next
command
 sed -i "s/^${PAGE_TITLE},.*$/${PAGE_TITLE}, ${PAGE_URL}/" $TABSFILE
fi
```

Note: If installing your own files, they must exist in a subdirectory of /home/httpd/jbmconfig/html. E.g.

```
/home/httpd/jbmconfig/html/mypage/
/home/httpd/jbmconfig/html/mypage/index.html
/home/httpd/jbmconfig/html/mypage/styles.css
```

After any changes are made, refresh the GUI page.

Install using a .zip package

This is how you would install a web page with your SDK package. First make sure all necessary files for your site are descendants of a single folder. E.g.

```
$ unzip -l mypackage.zip
install.sh
home/
home/httpd/
home/httpd/jbmconfig/
home/httpd/jbmconfig/html/
home/httpd/jbmconfig/html/mypage/
home/httpd/jbmconfig/html/mypage/index.html
home/httpd/jbmconfig/html/mypage/mystyle.css
home/httpd/jbmconfig/html/mypage/myjavascript.js
```

Note: the path "**home/httpd/jbmconfig/html**" is critical. Your folder "mypage" needs to extract here for the device to be able to serve it. Your SDK binaries can extract to the standard path.

This sample install.sh would install your package and create a tab in the GUI

```
#!/bin/bash

# This is the text for the button that will appear in the main navigation
drop-down
PAGE_TITLE="My Page"

# This is the name of the directory containing your files
#     (relative to the "html" directory in your package, e.g.
home/httpd/jbmconfig/html/mypage)
PAGE_URL="mypage"

# You should not need to touch anything below this line

ZIP_FILE="$1"
unzip -o "$ZIP_FILE" -x install.sh -d /

# This adds your page title to our list of custom URL's (but only if it
hasn't already)
if ! grep "^${PAGE_TITLE}" "/home/httpd/jbmconfig/txt/customTabs.txt"
&>/dev/null; then
 echo "${PAGE_TITLE}, ${PAGE_URL}" >>
"/home/httpd/jbmconfig/txt/customTabs.txt"
fi
```

You can then install **mypackage.zip** through the GUI, and the **Custom** tab should appear after a refresh.

Web UI Integration

This page details setting up your own interface to be served by our web server.

Quick Start

Use this if you already have your page well defined, and just want to drop in your files

Server Root

Place files here to have them available via direct URL

```
/home/httpd/jbmconfig/html/
```

Example

```
/home/httpd/jbmconfig/html/sdk/mypage.html
```

Will be available at

```
http://1.2.3.4:10000/sdk/mypage.html
```

CGI

For system level access to the device, you can put scripts in this folder:

```
/home/httpd/jbmconfig/cgi-bin
```

And you can then query them from your html/javascript.

Example

```
/home/httpd/jbmconfig/cgi-bin/mycgi.cgi
```

Can be hit via this jquery call:

```
$.post("cgi-bin/mycgi.cgi")
```

Integration

For more seamless integration with our primary Web UI, you can include a link to your page from our UI and even load your page using our javascript framework to gain our header, footer, and access to our jquery, knockout and GauWeb libraries

Extensions Tab

This tab is populated from an on-device configuration. To manually add your page, navigate to **Advanced** > **Expert Mode** > **Configure Sub-systems** in our Web UI and edit the **Custom Extensions Tab** file. There are examples included.

You can also do this automatically during package installation

Examples

AMD Integrated Page

Add the following to your install.sh for a link to your integrated page:

```
/usr/iog/api/webui_extension_add -p "My Page" "mypage"
```

As long as your package contains these files:

```
/home/httpd/jbmconfig/html/pages/mypage.html
/home/httpd/jbmconfig/html/pages/mypage.js
```

Your page will appear in our Extensions drop down with a link named **My Page** pointing to **http://1.2.3.4:10000/#!/mypage** (relative to your access method and device IP). See the appendix for more information

Independent Page

For a link to your fully independent page that does not use our pager js infrastructure, add the following to your install.sh for a direct URL:

```
/usr/iog/api/webui_extension_add "My Page" "sdk/mypage.html"
```

And as long as your package includes **`/home/httpd/jbmconfig/html/sdk/mypage.html`**, a link with the label **My Page** will appear in our extensions drop down.

Pager JS AMD Integration

Our primary Web UI uses pagerjs and requirejs for Asynchronous Module loading of content.

If you use **webui_extension_add** with the "**-p**" flag to include your page, you can use our javascript infrastructure by accessing your page via an #!/mypage URL. This gives your page easy access to our jquery/knockout and custom GauWeb libraries. See **Pager JS Integration** in the appendix for sample html/js files.

Appendix

install.sh

Functions to use in install.sh for package installation

webui_extension_add

Add your page to the Extensions drop-down tab. Usage:

```
/usr/iog/api/webui_extension_add [-p|-r] <Title> [URL]
```

-r	(default) Inject raw URL (path/to/mypage.html
-p	Use pagerjs URL (#!/mypage)
Title	The label for your link in the drop-down

URL	(optional, defaults to Title) URL link will point to. Potentially prepended by **-p** flag

File Locations

Path	Description
/home/httpd/jbmconfig/html	Server root. Places files here for access via direct URL
/home/httpd/jbmconfig/html/pages	PagerJS root. Place mypage.html and mypage.js here for access via AMD/PagerJS
/home/httpd/jbmconfig/cgi-bin	Path to on-device cgi scripts. Accessible via REST request at <host>:10000/cgi-bin/

Pager JS Integration

Expand on these files to integrate with our Pager JS infrastructure (remember, no html/head/body tags necessary)

```
/home/httpd/jbmconfig/html/pages/mypage.html:
<input type="text" data-bind="value: myinput"></input>
<button data-bind="click: randomize">Generate Random Number</button>

/home/httpd/jbmconfig/html/pages/mypage.js:
define(["jquery", "knockout", "GauWeb"], function ($, ko, GauWeb), function
() {return new function () {
 var self = this
 self.myinput = ko.observable("default value")
 self.randomize = function () {
  self.myinput( Math.random() )
 }
}})
```

GauWeb API

These functions are available in your page javascript if you load your page via our Pager JS infrastructure and include "GauWeb" in your module's definition (see Pager JS Integration example above)

GauWeb.ui

Functions to produce visible modals and promote user interaction

Available since version 4.24

GauWeb.ui.loader.show

This will make a loading modal appear with spinner gif. Optionally, call with a string argument to show a custom message.

GauWeb.ui.loader.hide

Hide the modal.

> The loader maintains a stack. For every instance of "loader.show" there must be an instance of "loader.hide." This allows nested CGI

calls to show/hide their own loader without making the loader disappear prematurely. This effectively changes the loader message based on the current action.

xfglib - multi-subsystem xml configurator

SYNOPSIS

```
require "/etc/jbm/xfglib.pl";
```

This script is a replacement for jbm_xmllib.pl that supports multiple-subsystem operations.

Build up a configuration in memory with repeated calls to "xfg_set_attribute"

```
&xfg_set_attribute($subsystem, $attribute, $value);
```

Each set with a matching $subsystem will apply the attribute to that subsystem's list of attributes

You can also specify multiple attributes in a hash with "xfg_set_multi_attribute"

```
 my %xml_settings = (
    dns1 => "8.8.8.8",
    eth1_enable => "Yes"
);
&xfg_set_multi_attribute("dhcpserver", \%xml_settings);
```

This invocation mimics jbm_xmllib.pl

Note: "action" does not need to be specified here. One action will apply to all changes at the end.

After you have added configuration options for all your subsystems, run the following with a standard action to commit the change

```
&xfg_commit( $action [, 1] );
```

Note: This will CLEAR all previously set configurations unless a second argument of 1 or "truthy" is passed in to preserve the in-memory config map

Quick jbm_xmllib.pl Conversion

Previous implementation using jmb_xmllib.pl - **3 calls** to migratecfg

```
 my %xml_settings = (
    action => "apply",
    dns1 => "8.8.8.8",
    eth1_enable => "Yes"
    );
&xmlcfg_dhcpserver(\%xml_settings);

%xml_settings = (
    action => "apply",
    enable => "y"
    );
&xmlcfg_firewall(\%xml_settings);

%xml_settings = (
    action => "apply",
    intfs => "eth0,usb0"
    );
&xmlcfg_firewall_trustedintfs_add(\%xml_settings);
```

Implement using xfglib.pl - **1 call** to migratecfg

```
my %xml_settings = (
            dns1 => "8.8.8.8",
            eth1_enable => "Yes"
            );
        &xfg_set_multi_attribute("dhcpserver", \%xml_settings);

&xfg_set_attribute("firewall", "enable", 'y');

&xfg_table_append("firewall", "trustedintf", {intf => eth0});
&xfg_table_append("firewall", "trustedintf", {intf => usb0});

&xfg_commit("apply");
```

FUNCTIONS

xfg_set_attribute

Arguments

1. $subsystem - string containing subsystem name
2. $attribute - string attribute name
3. $value - string for attribute value

Description

Set the value of subsystem attribute in memory. New compared to jbm_xmllib.pl, this is used for one-off sets for the attribute of a subsystem. It is the backbone for the "xfg_set_multi_attribute" subroutine.

xfg_set_multi_attribute

Arguments

1. $subsystem - string containing subsystem name
2. \%attributes - hash reference of attribute:value pairs

Description

For each attribute in the %attributes hash reference, set an associated value for subsystem

This is designed to mimic jbm_xmllib.pl's method of setting configurations

```
# jbm_xmllib.pl:
$ret = &xmlcfg_dhcpserver(\%xml_settings);

# xfglib.pl:
$ret = &xfg_set_multi_attribute("dhcpserver", \%xml_settings);
```

The same has structure of attribute:value, with exception of "action," is used to make direct conversions more straightforward. Action is specified in the call to "xfg_commit"

xfg_table_append

Arguments

1. $subsystem - string containing subsystem name
2. $tablename - string containing name of table
3. $tablehash - hash reference containing attribute:value pairs

Description

Set the value of subsystem attribute in memory. New compared to jbm_xmllib.pl, Similar to "xfg_set_multi_attribute", except the hash reference contains all attributes pertaining to a specific table record. This also mimics jbm_xmllib.pl usage:

```
# jbm_xmllib.pl:
&xmlcfg_firewall_trustedintfs_add(\%xml_settings);

# xfglib.pl:
xfglib: &xfg_table_append("firewall", "trustedintf", \%xml_settings);
```

xfg_table_clear

Arguments

1. $subsystem - string containing subsystem name
2. $tablename - string containing name of table

Description

This is used to explicitly clear a table in config.xml. Tables are replaced in whole chunks, so a clear is performed by replacing the list of records with an empty list.

xfg_clear

Description

Clear all in-memory configurations. This is used at the end of an "xfg_commit" **by default** to clear the slate for future configurations

xfg_commit

Arguments

1. $action - action string passed to migratecfg. E.g. "apply", "saveall", "cfgonly"
2. $preserve - (optional) call with a "1" as the second argument to preserve the in-memory set of configurations

> Use this if calling routine is a daemon that will want to repeatedly apply the same or similar set of configurations

Description

This is the main make-configuration-happen subroutine. New compared to jbm_xmllib.pl, this is the only subroutine that actually modifies config.xml. All other functions manage an in-memory set of configurations, "xfg_commit" is called once at the end to apply the configurations all at once.