

1. SN SDK Setup And Installation	2
2. SN-SDK User Guide	9
3. SN SDK IODB Library Reference	17
4. SDK How-To's	24
4.1 XFGLib User Guide	24
4.2 Controlling Cellular Connection	29
4.2.1 Lighttpd testing	32
4.3 SMS Messages	33
4.4 Sending Emails	42
4.5 System Statistics	46
4.6 Getting GPS data from the Device	50
4.7 AT Command Interface	58
5. Include Libraries	61
6. snupdate	65
7. Package Preservation	68
8. GAU Custom Extensions	70
8.1 Add Custom Tabs to Navigation	73
9. Web UI Integration	75
10. xfglib - multi-subsystem xml configurator	80

SN SDK Setup And Installation

- Introduction
 - Supported Build Host
- VM Installation
 - VirtualBox Installation
 - Ubuntu Installation
 - Ubuntu Setup
 - Screen resolution adjustment
 - Setting up shared folders
- SDK Installation
 - System Setup
 - Toolchain Installation
 - Bootstrap Installation
 - 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

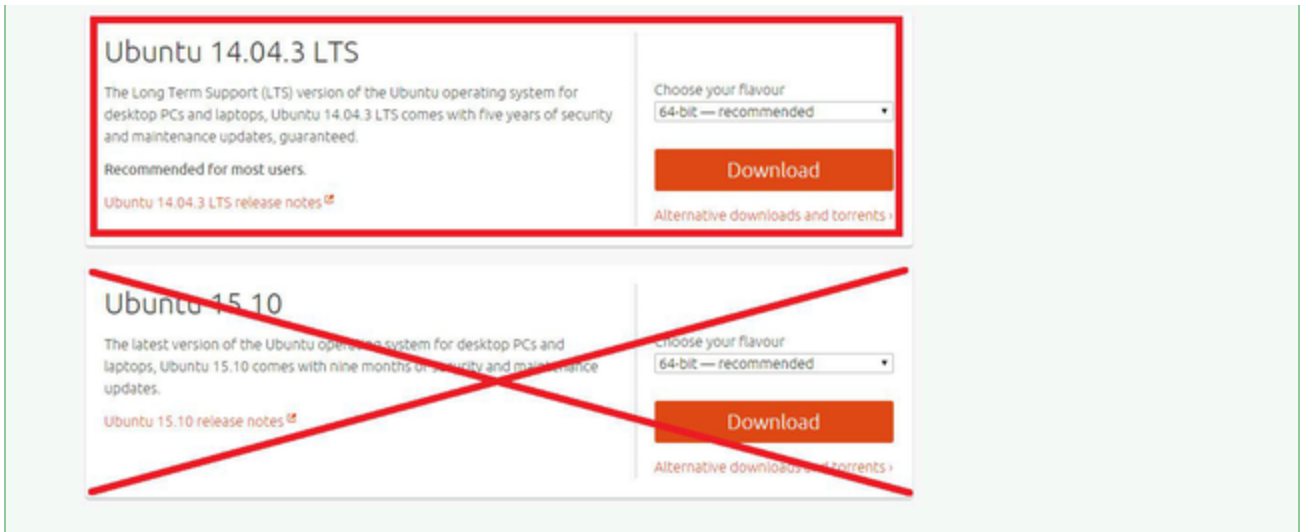
1. Download the latest version of VirtualBox from <https://www.virtualbox.org/wiki/Downloads>.
2. Download the Oracle VM Virtual Machine Extension Pack. The Extension Pack is for USB connections.
3. After installer has been downloaded, run the installer and install VirtualBox.
4. Once VirtualBox has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




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**.

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:

Type: 

Version:

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**.



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 disk 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)
- ☐ VHDX (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 saved. 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

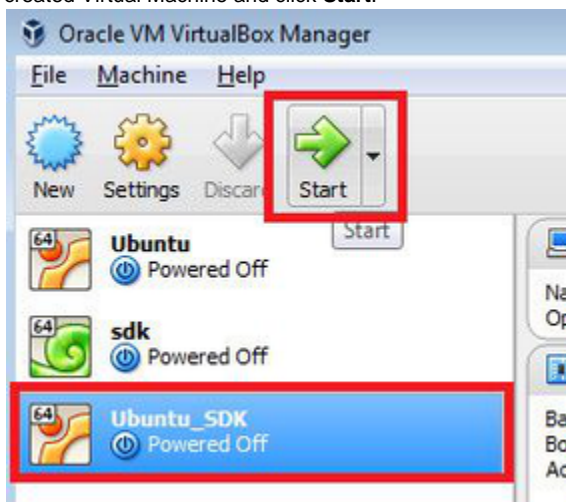
Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.

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.




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.

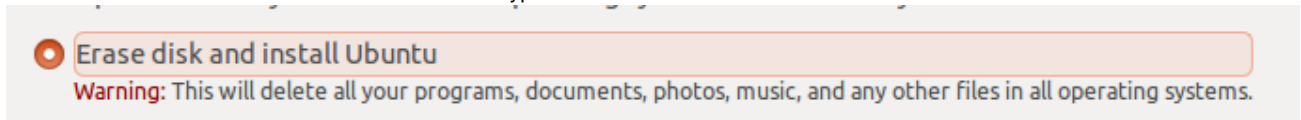
The disk should be suitable for starting a computer from and should contain the operating system you wish to install on the virtual machine if you want to do that now. The disk will be ejected from the virtual drive automatically next time you switch the virtual machine off, but you can also do this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

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

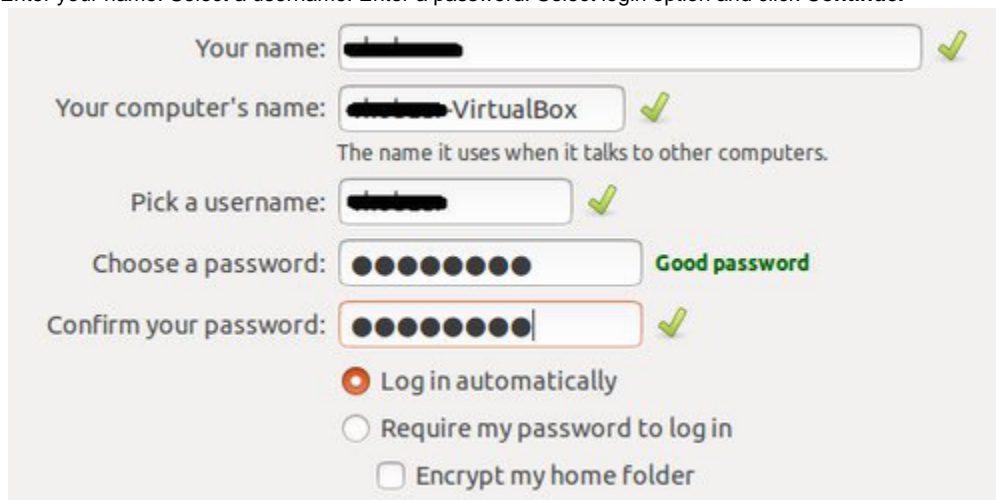
12. Ubuntu will start booting. On Ubuntu *Welcome* screen, Click **Install Ubuntu**.



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



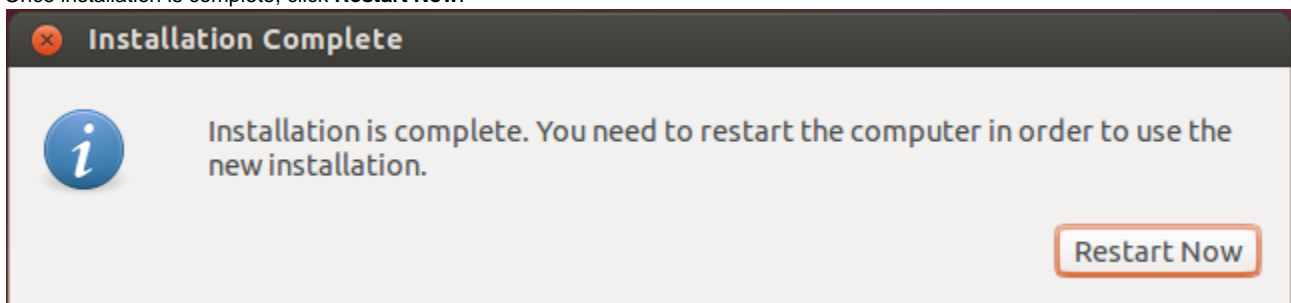
15. On *Write the changes to disk?* warning message, click **Continue**.
16. Select your timezone and click **Continue**.
17. Select your keyboard layout and click **Continue**.
18. Enter your name. Select a username. Enter a password. Select login option and click **Continue**.



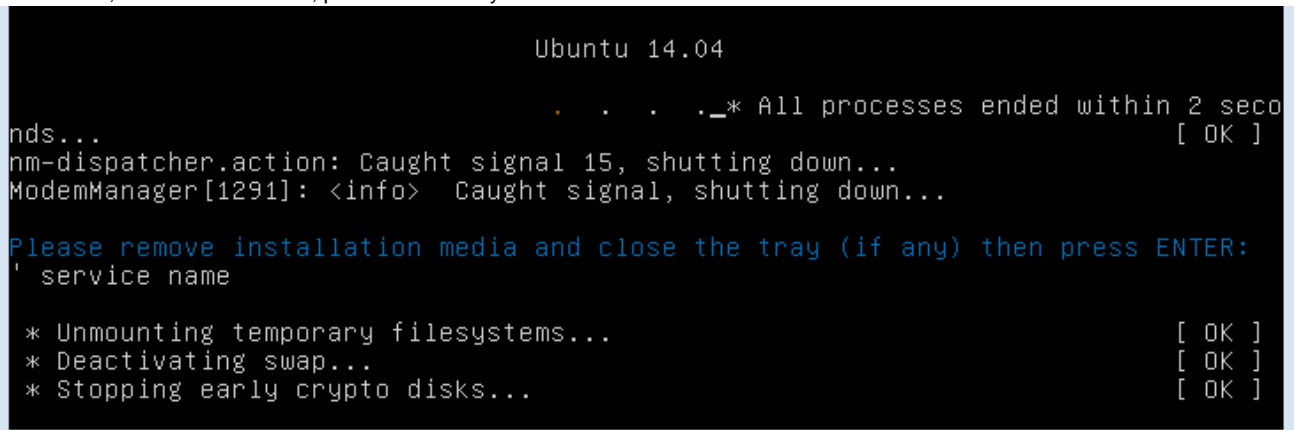
19. Installation is now in progress. This may take few minutes.



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



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



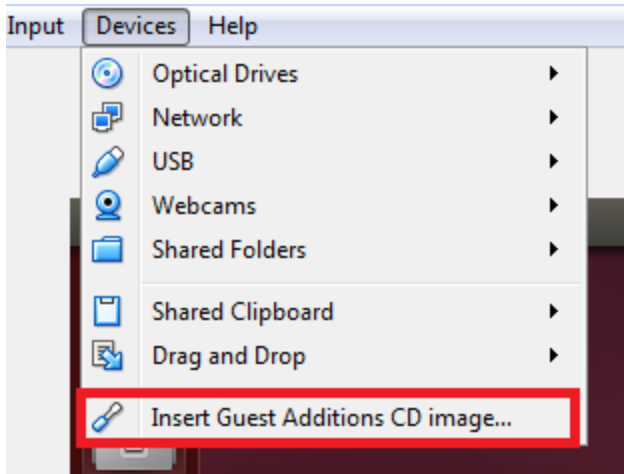
Ubuntu Setup

Screen resolution adjustment

1. On your 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

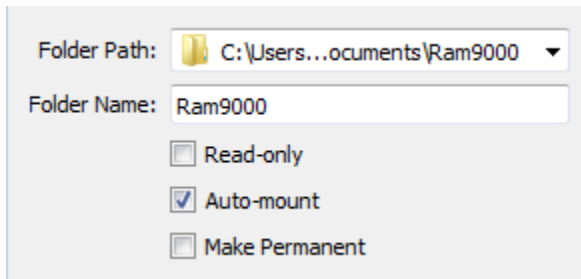
Following steps are to setup a shared folder on Windows OS that you could access from Ubuntu. This is necessary to be able to transfer files to a from Ubuntu.

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**



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>

```
sf_Ram9000@sf_Ram9000-VirtualBox:~$ cd /media
sf_Ram9000@sf_Ram9000-VirtualBox:/media$ ls
sf_Ram9000@sf_Ram9000-VirtualBox:/media$
```

[Top](#)

SDK Installation

System Setup

As the base ELDK toolchain 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
```

Toolchain Installation

The latest binary distribution contains both toolchain compilers and compiled-for-device overlay libraries with headers. This can be unpacked anywhere using tar:

```
tar -zxf snsdk.4.27.tgz -C <target directory>
```

See distribution package for more details.

[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 -zxvf snsdk_bootstrap_v4.tgz // Current Directory

// OR

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

[Top](#)

SN-SDK User Guide

- Building the Program
- Installing the Example Program
 - Product On-Board Web Interface Installation
 - Sixview Manager Installation
 - Command Line Interface
- Additional Details
 - Libraries
 - configure.sh
 - sdk.conf
 - Controlling Package Installations
 - Install.sh functions
 - Install.sh functions usage example
 - install.sh example file
 - Example Application Persistence
 - Factory Reset
- GAU Interaction with SDK Application
 - Sample custom SDK page
 - Step to enable SDK Sample Page
- 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 folder.

```
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

Change

Remove



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 successfully

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 verify firmware version and to extract 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.

Install.sh functions

Install.sh has number of functions that assist in verifying firmware versions and cpu types. These functions are included in the sample install.sh.

Warning: DO NOT modify these functions.

- **printver()** - Prints the current firmware version of the device

printver()

- **checkcpu()** - checks if the cpu of the device
Arguments: string cpu type. Valid cpu names are: g25, bt6k
Returns: 0 on match, 1 if no match

checkcpu()

- **checkcpu_bt6k()** - checks if CPU is a bt6k CPU9, exits on failure

checkcpu_bt6k()

- **checkcpu_g25()** - checks if CPU is a G25, exits on failure

```
checkcpu_g25()
```

- **checkcpus()** - Checks a list of CPU types against checkcpu()
Arguments: a list of cpu types space seperated. Ex: "g25 btk6"
Returns: 0 on match, 1 if no match

```
checkcpus()
```

- **checksn()** - Checks if the device is in the SN product line: btk6, g25
Returns: 0 on match, 1 if no match

```
checksn()
```

- **vercomp()** - compare between two versions
Arguments: arg1 - first version, arg2 - second version to compare to
Returns: 0 if both match, 1 if arg1 is greater than arg2, 2 if arg1 is less than arg2

```
vercomp()
```

- **checkversion()** - Checks and compare version of the device
Arguments: **Arg1** - max version - less than or equal to, **Arg2** - min version (optional) - greater than or equal to
Returns: 0 - match is ok, 1 - match is out of range (error)

```
checkversion()
```

Install.sh functions usage example

- The following line enables the init script for the program meaning it will start whenever the device boots.

```
chkconfig sdk on
```

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

```
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.

- Test if the current build version is 4.24 and below for an install

```

checkversion "4.24"
if [ $? = 1 ]; then
    printver
    echo "Package is for build 4.24 and lower, exiting"
    #exit 1
else
    echo "Version check OK for 4.24"
fi

```

- Test if the current build version between 4.24 and 4.20, higher version must be first

```

checkversion "4.23" "4.20"
if [ $? = 1 ]; then
    printver
    echo "Package is for build 4.20-4.23, exiting"
    #exit 1
else
    echo "Version check OK for 4.20-4.23"
fi

```

- Manual test for a G25

```

checkcpu "g25"
if [ $? = 1 ]; then
    echo "Error: Not a G25 CPU"
    #exit 1
fi

```

Also, see additional [snupdate](#) documentation for more information on creating package .zip files.

install.sh example file

install.sh must be saved in UNIX, not DOS format.
 If using Notepad++ to edit install.sh file, make following selection: Edit EOL Conversion UNIX/OSX Format

install.sh

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](#)

Factory Reset

If your package is installed to persist across firmware upgrades, a factory reset will cause your package to be re-installed. If your application creates any files at run-time post-install, these files will not be touched by the factory reset process.

To have run-time files removed by a factory reset, put them here:

```
/vault/appdata
```

This folder is removed and recreated during a factory reset

This feature is only available on select models. Check for directory presence before writing any files

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 your 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)

Start

Stop

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/jbmconfig/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

SN SDK IODB Library Reference

- [IODB Functions](#)
 - [IODBRead](#)
 - [IODBWrite](#)
 - [IODBWriteMask](#)
 - [IODBVersion](#)
- [File List](#)
 - [iodb.h File 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 available 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	Definition
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 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 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 significant 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 EIOBNOTFOUND 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 IOB 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 IOBerr unsigned short
```

Functions

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

IOBerr IOBWrite (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.

IOBerr IOBWriteMask (USHORT, USHORT, USHORT, void , BYTE , const char
)

USHORT IOBVersion (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 IOB 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 ? Required

Listening IP Port: 22 ? Required

Login Grace Time (seconds): 120 ? Required

Maximum Concurrent Connections: 20 ? Required

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
- Example Output
 - Status of Cell Modem Up
 - Status of Cell Modem Off/Down
- Cellular statistics

[illegible]

'stop' note

After issuing 'stop' all modem control and data connections stop. Modem will reset after 15 minutes via cron job if left in this state unless tempoff or permoff has been issued

Modem resets are only allowed once every 5 minutes. If a modem command set of 'start, stop, start' are issued in less than 5 minutes, the modem may not start back up immediately, but within a delayed 15 minutes.

cellmodemconnect.pl

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

Options	Information	SMS/RSSI Status	After a reboot	System Config
Usage	<code>cellmodemconnect.pl <start con nocon clear stop tempoff permoff on reset status></code>			
start	Issues a modem reset. 'stop' must be called first.	Paused during reset	Returns to config setting	Unaffected
stop	Stop cell connection and polling programs (backends)	Stopped. No signal updates, no SMS	Returns to config setting	Unaffected
tempoff	Power off Cellmodem until 'on' or rebooted	Stopped. No signal updates, no SMS	Returns to config setting	Unaffected
permoff	Power off Cellmodem, turn off in config	Stopped. No signal updates, no SMS	Stays off	Forced to disable
on	Power on Cellmodem, Enable interface in config	Enabled	Enabled	Forced to enable
con	Signal backend processes to attempt a data connection, temporarily overriding config.	Unaffected	Returns to config setting	Unaffected
nocon	Signal backend processes to stop a data connection, temporarily overriding config.	Unaffected	Returns to config setting	Unaffected
clear	Clear both con/nocon triggers. Device should go back to config setting	Unaffected	Returns to config setting	Unaffected
status	Report current connection status	Unaffected	Unaffected	Unaffected
reset	Hard modem reset	Paused during reset	Unaffected	Unaffected

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

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

cellmodemconnect.pl v1.6

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

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

stop     - Stop cell connection and polling programs (backends)

tempoff  - Power off Cellmodem until 'on' or rebooted

permoff  - Power off Cellmodem, turn off in config

on       - Power on Cellmodem, Enable interface in config

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

Notes: 'start' or 'reset' may reboot unit depending on modem.
       'permoff' and 'on' will make config changes when called from the
command line
```

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.

Lighttpd testing

When [lighttpd](#) source code is updated and custom patched (to include system authentication) it needs to be tested.

Current version 1.4.45 (.1)

Previous version source was modified/changed to upload large files into one big file chunk in `/tmp/`. The reason is, at the time, the source would not clean up the small file pieces as it globbed them back together in order, resulting in double-memory being used until the last chunk was pieced into the main file final name. This also resulted in the gau source and cgi scripts needing to be custom modified to handle the custom file code.

Now the current version removes the pieces as needed, and manages memory much more efficiently. So the gau source and cgi scripts can use "normal" file operating procedures to handle the files.

GAU Pages

A sample of some of the gau pages that need to be tested with the latest lighttpd code.

Upload Pages :

Package Install - verify packages upload and install properly

Firmware Update - Verify firmware images upload and apply properly

Certificate Manager - Verify certificates upload and install correctly

Download Pages :

Data Logger

Gwlnx log files

Gatherstats / Gatherconfigs

SD Card MAnager (Sentry)

Upload / Download Pages :

Configuration Manager

Other pages with upload and or download capabilities ?

Batch IO page?

IO Register page ?

Test IO page ?

Other pages

Admin Access - test http / https switching

SN Proxy Settings - test proxy (which is a separate lighttpd process)

SMS Messages

- [Overview](#)
 - [Compatible Sierra Modules](#)
 - [SMS Directory Structure](#)
 - [Process SMS file order example](#)
 - [SMS Character set](#)
- [Sending SMS Messages](#)
 - [Format for SMS file for sending :](#)
 - [Example](#)
 - [Output](#)
 - [Success log output](#)
 - [Failed log output](#)
- [Receiving SMS Messages](#)
 - [File format of received SMS Message :](#)
 - [Example Output](#)
- [Testing Long SMS](#)
 - [Incoming SMS](#)
 - [Test Outgoing Basic ASCII Character Set](#)
- [Supported character sets](#)
 - [RAM-99xx/RAM-69xx](#)
 - [All others](#)
 - [Encoding issues:](#)
- [Addendums](#)

Basic Instructions for Sending and Receiving of SMS Messages 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_parts/` - Directory to hold SMS messages that are part of a multi-part message. These messages will either be reassembled and placed in `/tmp/sms/recv/` when all of the parts arrive, or removed after 30 mins

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

`/tmp/sms/recv_post/` - A Directory to hold messages after they have been processed from `/tmp/sms/recv/`. An external SMS processor should move the file here when finished.

Outside processes **SHOULD NOT** process files in `/tmp/sms/recv_parts/`

Process SMS file order example

1. SMS arrives in `/tmp/sms/recv/sms_msg_6185551212_20170626_095051.1`
2. External parser processes SMS file `sms_msg_6185551212_20170626_095051.1`
3. External process finishes processing file and moves it to `/tmp/sms/recv_post/`
4. Cell-modem control process checks `/tmp/sms/recv_post/` for files older than 24 hours and deletes them

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

SMS Messages in `/tmp/sms/recv_parts/` will be removed after processing or the age is 30 mins

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 918. Messages larger than 918 2characters will be truncated.
- Fields must be on separate lines.

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=8005551212, 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_**PhoneNumber**_YYYYMMDD-hhmmss.x

Where

- **PhoneNumber** = phone number
- **YYYY** = Year
- **MM** = Month
- **DD** = Day
- **hh** = Hour
- **mm** = Minute
- **ss** = Second
- **x** = an incremental digit that goes from 1 to 65535 and then starts over, not guaranteed to increment by 1 each time

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 text until the end of the file, including <LF>/<CR>

Example Output

Below is a system log of SMS Message received

```
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
```

Testing Long SMS

Some long SMS is supported, but not for all carriers.

Incoming SMS

Long incoming SMS should be supported on all carriers currently, except for Verizon. This may change in the future.

This example will show how a long incoming SMS might look on an AT&T account. Long SMS messages can contain up to 918 characters.

Sending the following text to a device:

```
This example illustrates how to write plain text in an HTML file. Blank
lines (as next line) are ignored. Similarly, if you press the Enter key,
you will not get a new paragraph. To illustrate this I am pressing Enter
right here ... and as you can see, the line continues. If you want to break
the line you need to enter the "P" tag, like right here ...
```

Will be broken up into 3 separate messages, and resembled on the device. Here is a sample log. Notice part 2 arrived, then part 1 and 3.

```
Jul 18 12:22:01 generic_watch[4455]: Received SMS Part 2/3 (ID: 73) from
8005551212 at 18/07/17 - 12:22:01, Message: not get a new paragraph. To
illustrate this I am pressing Enter right here ... and as you can see, the
line continue. If you want to break the line you
Jul 18 12:22:07 generic_watch[4455]: Received SMS Part 1/3 (ID: 73) from
8005551212 at 18/07/17 - 12:22:03, Message: This example illustrates how to
write plain text in an HTML file. Blank lines (as next line) are ignored.
Similarly, if you press the Enter key, you will
Jul 18 12:22:07 generic_watch[4455]: Received SMS Part 3/3 (ID: 73) from
8005551212 at 18/07/17 - 12:22:04, Message: need to enter the "P" tag, like
right here ...
Jul 18 12:22:09 generic_watch[4455]: Received SMS from 8005551212 at
18/07/17 - 12:22:03, Message: This example illustrates how to write plain
text in an HTML file. Blank lines (as next line) are ignored. Similarly, if
you press the Enter key, you will not get a new paragraph. To illustrate
this I am pressing Enter right here ... and as you can see, the line
continue. If you want to break the line you need to enter the "P" tag, like
right here ...
```

The message should be reassembled correctly when all the parts are present:

```
[root@RAM-123abc tmp]# cat
/tmp/sms/recv/sms_msg_8005551212_20170718-122209.1
FROM=8005551212
TIME=18/07/17 - 12:22:03
MSG=This example illustrates how to write plain text in an HTML file. Blank
lines (as next line) are ignored. Similarly, if you press the Enter key,
you will not get a new paragraph. To illustrate this I am pressing Enter
right here ... and as you can see, the line continue. If you want to break
the line you need to enter the "P" tag, like right here ...
[root@RAM-066e4f test_bad_parts]#
```

Test Outgoing Basic ASCII Character Set

Send from a device to a phone device. Paste the following into a file called /tmp/test.sms and replace <phone number> with the destination number


```
TO=<phone number>
MSG=ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
123456789
! @ # $ % ^ & * ( ) _ +
- =
[ ] \
{ } |
; '
: "
, . /
< > ?
```

Copy the the file into the send directoy and observe the system log :

```
[root@RAM-066e4f tmp]# cp test.sms /tmp/sms/send/
[root@RAM-066e4f tmp]# cat /var/log/messages| grep "Send Success"
Aug 12 09:08:47 generic_watch[5026]: Send Success: TO=8005551212,
MSG=ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 123456789 ! @ # $
% ^ & * ( ) _ + - = [ ] \ { } | ; ' : " , . / < > ?
[root@RAM-066e4f tmp]#
```

The the recipient phone and verify that every character sent looks correct.

Supported character sets

RAM-99xx/RAM-69xx

- 7bit GSM Encoding/Decoding
- 8bit Encoding/Decoding

Even though all characters can be encoded/decoded, it does not necessarily mean the character set is supported by the SMS processing engine

All others

- 7bit GSM Encoding/Decoding
- 8bit Decoding (inbound) only

For 7bit decoding, inbound characters with a hex value greater than 0x7F, including Greek letters, will be converted to a '?'

For 8bit decoding, inbound characters, Basic Latin ASCII and Latin-1 Supplement will be allowed. Other characters will be converted to a '?'

Encoding issues:

It has been noted on **BELL/Canada** that characters in the ASCII Basic Character Set Extension which are escaped and in 7-BIT encoding will be changed by the network to their ASCII counterparts. That is '[' becomes '<', ']' becomes '>'. There is not current work around for this. It maybe fixed in the future with USC2 encoding.

Addendums

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

1. If a unit reboots or loses 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
- Configuring the Device to Send Emails
 - Configuring with GUI
 - Configuring with XFGlib
- Email Message Format
 - Email Message File Format
 - Message body length
 - Email Character set
- Sending Emails
 - Example
 - Output
 - Success log output
 - Unsuccessful log output
- Sending Emails with Attachments
 - email_attachment
 - Attachment Types
 - Example

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 UNKNOWN
```

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> <PathToAnotherAttachment> [-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/emailMessage.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 emailMessage.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 is used to determine if 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
```

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 eth0 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

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
- GPS data from SNMP MIB
 - Requirements to obtain GPS data from SNMP MIB
 - List of supported MIBS

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++ compiler, 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 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
```

GPS data from SNMP MIB

GPS information can be obtained from your MIB Browser if the SNMP Agent is enabled on the device via GUI **Services > SNMP Agent**.

Requirements to obtain GPS data from SNMP MIB

1. Enable the SNMP Agent on the device by **Applying** the settings.
2. Download MIB to your PC by clicking **Download MIB** button.
3. Load the downloaded MIB to your **MIB Browser** and set up the device IP address.
4. Select gpscurrent from your MIB Browser **MIB Tree > iso.org.dod.internet > private > enterprises > redlionram > system** tree.

List of supported MIBS

Name	CurrentGpsValid
OID	.1.2.6.1.4.1.1890.1.7.1
MIB	RED-LION-RAM-MIB
Syntax	DISPLAYSTRING
Access	read-only
Status	mandatory
DefVal	
Indexes	
Descr	GPS Current Valid Fixed Quality (0 = Invalid, 1 = Valid)

Name	CurrentGpsLat
OID	.1.2.6.1.4.1.1890.1.7.2
MIB	RED-LION-RAM-MIB
Syntax	DISPLAYSTRING
Access	read-only
Status	mandatory
DefVal	
Indexes	
Descr	GPS Current Latitude Degrees

Name	CurrentGpsLong
OID	.1.2.6.1.4.1.1890.1.7.3
MIB	RED-LION-RAM-MIB
Syntax	DISPLAYSTRING
Access	read-only
Status	mandatory
DefVal	
Indexes	
Descr	GPS Current Longitude Degrees

Name	CurrentGpsAlt
OID	.1.2.6.1.4.1.1890.1.7.4
MIB	RED-LION-RAM-MIB
Syntax	DISPLAYSTRING
Access	read-only
Status	mandatory
DefVal	
Indexes	
Descr	GPS Current Altitude Tenths of Meter (280.2 = 2802)

Name	CurrentGpsTimeStamp
OID	.1.2.6.1.4.1.1890.1.7.5
MIB	RED-LION-RAM-MIB
Syntax	DISPLAYSTRING
Access	read-only
Status	mandatory
DefVal	
Indexes	
Descr	GPS Current Time Stamp

Name	CurrentGpsNumSat
OID	.1.2.6.1.4.1.1890.1.7.6
MIB	RED-LION-RAM-MIB
Syntax	DISPLAYSTRING
Access	read-only
Status	mandatory
DefVal	
Indexes	
Descr	GPS Current Number of Satellites

Name	CurrentGpsFtfromcp
OID	.1.2.6.1.4.1.1890.1.7.7
MIB	RED-LION-RAM-MIB
Syntax	DISPLAYSTRING
Access	read-only
Status	mandatory
DefVal	
Indexes	
Descr	GPS Current Feet From Lockdown Center Point

Name	CurrentGpsSpeed
OID	.1.2.6.1.4.1.1890.1.7.8
MIB	RED-LION-RAM-MIB
Syntax	DISPLAYSTRING
Access	read-only
Status	mandatory
DefVal	
Indexes	
Descr	GPS Current Speed, SOG tenths of knots (50.1 = 501)

Name	CurrentGpsCourse
OID	.1.2.6.1.4.1.1890.1.7.9
MIB	RED-LION-RAM-MIB
Syntax	DISPLAYSTRING
Access	read-only
Status	mandatory
DefVal	
Indexes	
Descr	GPS Current Course, Heading in tenths of degree (280.3 = 2803)

Name	GpsSource
OID	.1.2.6.1.4.1.1890.1.7.10
MIB	RED-LION-RAM-MIB
Syntax	DISPLAYSTRING
Access	read-only
Status	mandatory
DefVal	
Indexes	
Descr	GPS Source of Data (1 = Internet; 3 = Fixed)

Name	GpsLockdownState
OID	.1.2.6.1.4.1.1890.1.7.11
MIB	RED-LION-RAM-MIB
Syntax	DISPLAYSTRING
Access	read-only
Status	mandatory
DefVal	
Indexes	
Descr	GPS Current Lockdown State (0 = Monitor; 5 = Lockdown; 7-9 = Violation)

Name	GpsLockdownRadius
OID	.1.2.6.1.4.1.1890.1.7.12
MIB	RED-LION-RAM-MIB
Syntax	DISPLAYSTRING
Access	read-only
Status	mandatory
DefVal	
Indexes	
Descr	GPS Current Lockdown Radius (ft), Units in Feet as calculated from centerpoint

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: J11111111111111
+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: 1111111111111111
ESN: 1111111111, 80FFFFFF
IMEI: 1111111111111111
IMEI SV: 13
FSN: J1111111111111111
+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 include 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 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
libftd11.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
```

```
libnfnetworklink.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
libtinysql.so -> libtinysql.so.1.0.1
libtinysql.so.1 -> libtinysql.so.1.0.1
libtinysql.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
libsxiiodbc++.so -> libsxiiodbc++.so.1.3.2
libsxiiodbc++.so.1 -> libsxiiodbc++.so.1.3.2
libsxiiodbc++.so.1.3.2
libtimerc++.so -> libtimerc++.so.1
libtimerc++.so.1
libtokenstrtlutalec++.so -> libtokenstrtlutalec++.so.1
libtokenstrtlutalec++.so.1
```

snpupdate

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

- [Rules for Packages](#)
- [Modes of snupdate operation](#)
 - [Unscripted Installation Packages](#)
 - [Scripted Installation Packages](#)
 - [Sample install.sh](#)
 - [Configuration Only Packages \(gatherconfigs\)](#)
 - [Return Codes](#)
- [Examples of Common Tasks in Scripted Installs](#)
 - [Check Firmware Version](#)

Rules for Packages

- All package names must end in .zip

- snupdate will only work if the package file is 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.

Return Codes

Return codes are multiplied by 256, so a return code of 512 is actually a return code of 2. This is an effect of snupdate.

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

- [Partitions](#)

Package Preserve Automation

- [Usage](#)

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. Its 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+
/vault/appdata	Directory in /vault. Will be cleared if reflashed without saving configs or if Factory Reset is chosen	4.25

Take care when saving data to the file system. A program can open a handle to a file and start writing, but if the handle is not flushed or closed, the data could be kept in buffers that have not been written to "disk" (for example: /vault). If power off or reboot happens, then those buffers could be lost.

When writing to a file handle, always flush after writes to ensure data is kept. See Linux documentation on `fflush()`, `fsync()`, and `O_SYNC` flags for more information.

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
nvrnm (Non-Volatile Random Access Memory)	memory				Reads and Writes that can be retentive as long as the battery is charged. (Only Available on RAM 9000 series)

Based on the table above, the Vault partition is the best place to preserve a package. Its data will be preserved across both re-flash and advanced re-flash. By saving dynamic information into /vault/appdata, your custom configs can survive a reflash but will be cleared when the system does a factory reset. If you save your configs in /vault/mycustomdir/ then you will have to manage your own config status.

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.

User edits to system files are usually not preserved (for example, /etc/crontab). This is intentional.

/opt is not preserved across unit reflashes. Data kept in /opt, and other system changes on the root / partition, must be encapsulated into your installer. Make sure to preserve your package for re-installation across firmware reflashes (see below).

/vault/appdata will be cleared when a unit is returned to factory defaults. This is a useful area for your custom SDK configs and data that will be preserved across a firmware reflash. See the section "**Factory Reset**" for more information.

Package Preserve Automation

Package Persevere Automation will be available in version 4.24 and later

It is intended that your SDK package will be reinstalled freshly after every firmware reflash. Config changes and other dynamic data should be saved in /vault/appdata/ to persist reflashes, but still allow the system to clear them with a factory reset.

To ensure that an 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 if 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.

```
/usr/iog/api/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

```
/usr/iog/api/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

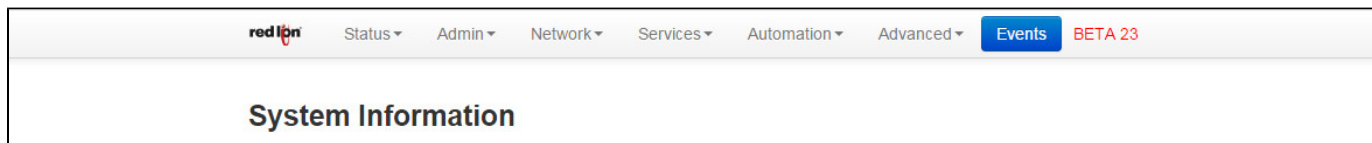
- [General Information](#)
- [Usage](#)
- [SDK Installation Implementation](#)
- [Notes](#)
- [Custom tabs on version 4.23.](#)

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

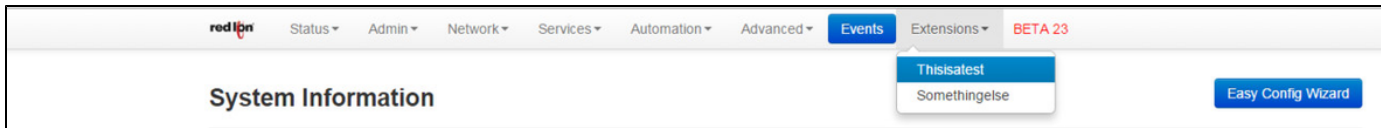
General Information

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](#)
 - [Server Root](#)

- Example
- CGI
 - Example
- Integration
 - Extensions Tab
 - Examples
 - AMD Integrated Page
 - Independent Page
 - Pager JS AMD Integration
- Appendix
 - install.sh
 - webui_extension_add
 - File Locations
 - Pager JS Integration
 - GauWeb API
 - GauWeb.ifconfig
 - GauWeb.ifconfig.[interface]
 - GauWeb.ifconfig.cell
 - GauWeb.ifconfig.refresh(callback)
 - GauWeb.ui
 - GauWeb.ui.loader.show
 - GauWeb.ui.loader.hide

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.ifconfig

Available since version 4.22

Contains information about active network interfaces

GauWeb.ifconfig.[interface]

Available interfaces: eth0, eth1, ppp0, usb0, wlan0, wwan0.

For each interface, the following attributes are available as **knockout observables**:

hwaddr	Hardware MAC Address
ipaddr	IPv4 Address
ip6addr	IPv6 Address
bcast	ipv4 Broadcast Address
mask	ipv4 Netmask
state	link state: UP or DOWN

To get the value, call the attribute as a function. (for more information, see knockoutjs documentation for observables)

```

GauWeb.ifconfig.eth0.ipaddr();
// returns "192.168.0.1"

```

Ensure you have called a refresh, as below, before trying to get the value of any attribute

GauWeb.ifconfig.cell

This is a special interface that assumes the attributes above for either `wwan0` or `ppp0`, depending on which interface is active.

Additionally, `GauWeb.ifconfig.cell.active()` will return `wwan0` or `ppp0` based on which interface is active.

GauWeb.ifconfig.refresh(callback)

Refresh the data in the ifconfig module. Callback is a function to be run after the device has responded. Example:

```
GauWeb.ifconfig.refresh(function () {  
    console.log("Everything up-to-date!");  
});
```

callback is **optional**. Since all ifconfig interface attributes are knockout observables, any subscribers will automatically be notified if their values change. See knockoutjs documentation for more information.

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.

Available since version 4.24

xfglib - multi-subsystem xml configurator

- SYNOPSIS
 - [Quick jbm_xmllib.pl Conversion](#)
- FUNCTIONS
 - [xfg_set_attribute](#)
 - [xfg_set_multi_attribute](#)
 - [xfg_table_append](#)
 - [xfg_table_clear](#)
 - [xfg_clear](#)
 - [xfg_commit](#)

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 jbm_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 = &xmllcfg_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:
&xmllcfg_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.

Example Walkthrough

Please see: [XFGLib User Guide](#)