User Guide

Keysight N6841A Embedded Applications Guide

KEYSIGHT TECHNOLOGIES

Notices

Copyright Notice

Manual Part Number

N6841-90005

Edition

Edition 1.3, April 2018

Published by:

Keysight Technologies, Inc.
1400 Fountaingrove Parkway
Santa Rosa, CA 95403

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Declaration of Conformity

Declarations of Conformity for this product and for other Keysight product may be downloaded from the Web. Go to "Declarations of Conformity" at http://www.keysight.com/go/conformity. You can then search by product number to find the latest Declaration of Conformity.

U.S. Government Rights

The Software is "commercial computer software," as defined by Federal Acquisition Regulation ("FAR") 2.101. Pursuant to FAR 12.212 and 27.405-3 and Department of Defense FAR Supplement ("DFARS") 227.7202, the U.S. government acquires commercial computer software under the same terms by which the software is customarily provided to the public. Accordingly, Keysight provides the Software to U.S. government customers under its standard commercial license, which is embodied in its End User License Agreement (EULA), a copy of which can be found at http://www.keysight.com/find/sweula. The license set forth in the EULA represents the exclusive authority by which the U.S. government may use, modify, distribute, or disclose the Software. The EULA and the license set forth therein, does not require or permit, among other things, that Keysight: (1) Furnish technical information related to commercial computer software or commercial computer software documentation that is not customarily provided to the public; or (2) Relinquish to, or otherwise provide, the government rights in excess of these rights customarily provided to the public to use, modify, reproduce, release, perform, display, or disclose commercial computer software or commercial computer software documentation. No additional government requirements beyond those set forth in the EULA shall apply, except to the extent that those terms, rights, or licenses are explicitly required from all providers of commercial computer software pursuant to the FAR and the DFARS and are set forth specifically in writing elsewhere in the EULA. Keysight shall be under no obligation to update, revise or otherwise modify the Software. With respect to any technical data as defined by FAR 2.101, pursuant to FAR 12.211 and 27.404.2 and DFARS 227.7102, the U.S. government acquires no greater than Limited Rights as defined in FAR 27.401 or DFAR 227.7103-5 (c), as applicable in any technical data.

Warranty

THE MATERIAL CONTAINED IN THIS DOCUMENT IS PROVIDED "AS IS," AND IS SUBJECT TO BEING CHANGED WITHOUT NOTICE IN FUTURE EDITIONS. FURTHER, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, KEYSIGHT TECHNOLOGIES DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. KEYSIGHT TECHNOLOGIES SHALL NOT BE LIABLE FOR ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, USE, OR PERFORMANCE OF THIS DOCUMENT OR OF ANY OF THE PRODUCTS TO WHICH IT PERTAINS. SHOULD KEYSIGHT TECHNOLOGIES HAVE A WRITTEN CONTRACT WITH THE USER AND SHOULD ANY OF THE CONTRACT TERMS CONFLICT WITH THESE TERMS THE CONTRACT TERMS SHALL CONTROL.

Safety Information

CAUTION

A CAUTION notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a CAUTION notice until the indicated conditions are fully understood and met

WARNING

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Edition History

Edition	Summary of Change	Written by	Date
1.3	Keysight standards update	D. Carpenter	2018-04-24
1.2	Re-formatted/edited	D. Carpenter	2018-01-03
1.1	Re-formatted/edited	D. Szpotanski	2016-12-09
1.0	Initial Version	L. Yue	2016-01–07

Contents

Introduction

An embedded App is a Linux x86 SAL (Sensor Access Library) application or shell script that can be run directly inside the N6841A RF Sensor to perform custom tasks. For example:

- Acquire real-time FFT data and discard unwanted readings.
- Continuously log the sensor's GPS location to a file while in a moving vehicle.
- Continuously ping an IP address on a router every X seconds so the router can keep a VPN connection open.

Sections that follow will describe the steps needed to run an embedded application or shell script inside the sensor.

Create an application

Compile a Linux x86 SAL application

Linux x86 SAL App prerequisites:

The 2.0.6 SMS installed on a Windows PC, for SAL API documentation and the Linux shared libraries.

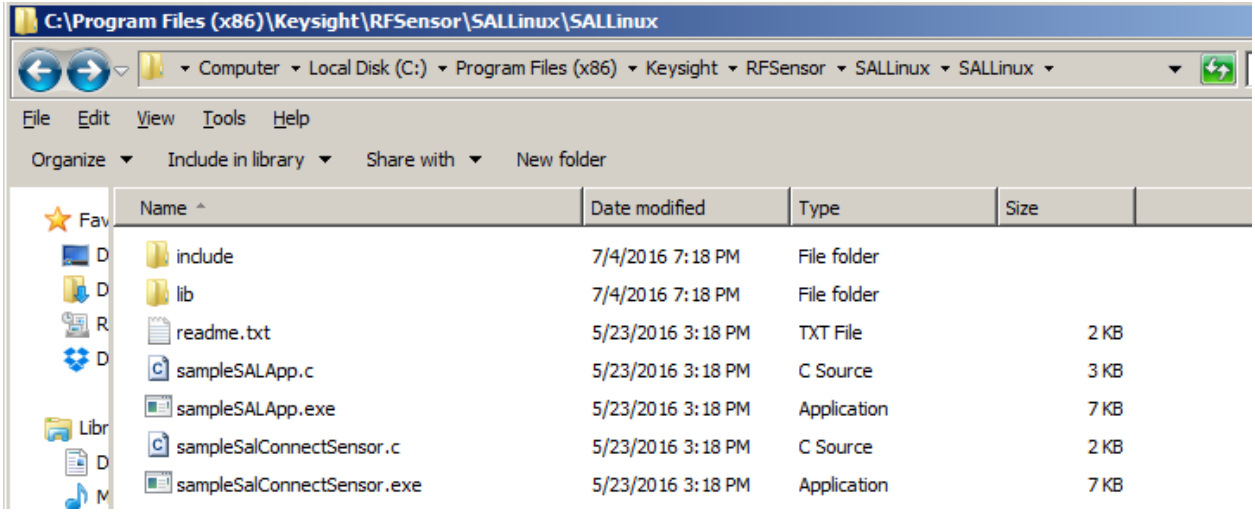
A Linux OS to compile and run the app. A 64-bit Linux OS is recommended instead of a 32-bit OS. (the 64-bit OS is more futureproof, although the SAL library will also work on a 32-bit OS).

Example OS: Red Hat Enterprise Linux/CentOS release 5.11 (Tikanga)

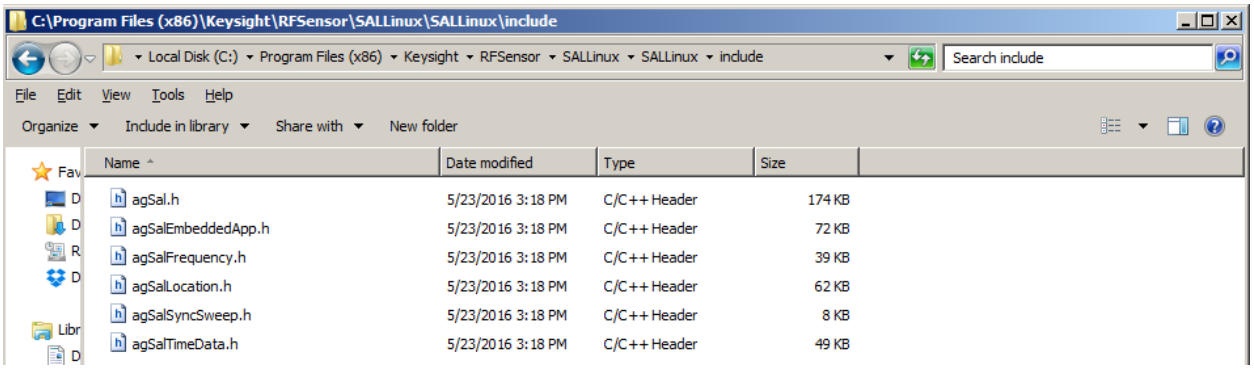
The SAL Linux files are located in a zip file on Windows:

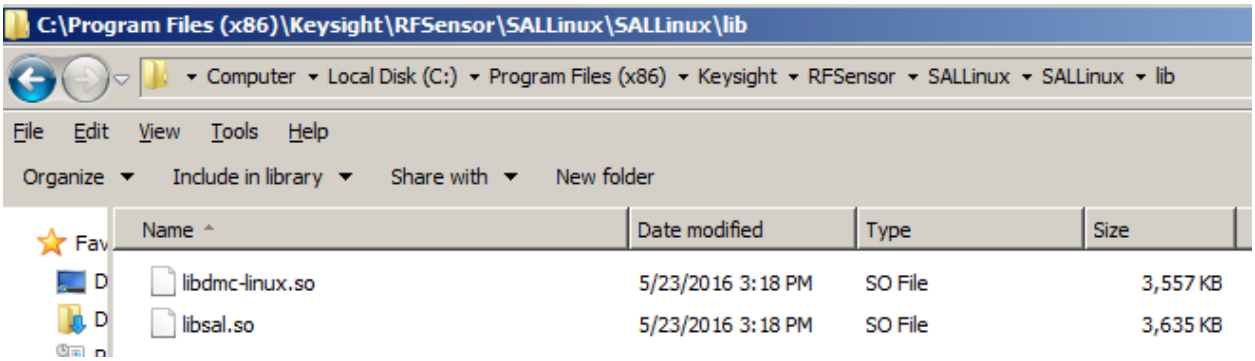
C:\Program Files (x86)\Keysight\RFSensor\SALLinux\SALLinux_2.0.6.xxxx.zip

When unzipped, a directory called "SALLinux" should contain a readme.txt and a couple of .c source code files. The readme.txt contains instructions on how to compile and run a SAL app in Linux using the sample source files.



The SAL Linux files are the header files in the "include" directory and the shared libraries in the "lib" directory. SAL header files and .so files need to be copied to a new Linux 32-bit app project, and their location added to include and library paths when invoking a compiler/linker.





The two .c source code files contain examples on how to open a connection to the Sensor Management Server (SMS) and get a list of sensors from the SMS, and how to open a connection directly to a sensor to get the sensor's current latitude/longitude location, and how to release the handles to prevent memory leaks.

The SAL API documentation can be found here:

Start -> All Programs > Keysight RF Sensor -> Documentation -> SAL Reference

C:\Program Files (x86)\Keysight\RFSensor\Manuals\agSalHelp.chm

Difference between embedded and host SAL Apps

Since an embedded SAL app is running directly inside the sensor instead of on a Linux PC, there are a few differences that need to be taken into account:

When invoking salConnectSensor3() to connect to the sensor, host IP address "127.0.0.1" should be specified.

Before writing to the USB drive, a check must be made to ensure the USB drive is present in the sensor. If salUSBls() with USB root directory "/" succeeds, the USB drive exists and can have files read from/written to.

The USB drive is mounted onto the Linux file system at the directory /data. The files and directories created must be placed inside "/data".

It is strongly recommended to call sync() to ensure that the data is properly flushed from the OS cache and committed to the USB drive. This is especially important if the sensor's power cable can be unplugged at any time.

Example code:

```
FILE* pFILE = fopen("/data/MyFileName.txt", "a");
if (pFILE)
{
   fprintf(pFILE, "Hello Sensor USB Drive World!\n");
   fclose(pFILE);
   pFILE = NULL;

   //Ensure that any data written to the USB drive is committed to
the drive.
   sync();
}
```

An example of an embedded SAL app can be found here:

C:\Program Files (x86)\Keysight\RFSensor\Embedded Apps Examples\GPSLocationLogger

Create a shell script

In some cases, the task to be performed is simple enough for the built-in shell commands inside the sensor's Linux OS. For example, the sensor might need to continuously ping an IP address on a router every X seconds, so the router can keep a VPN connection open. In this case, a simple "sh" shell script can be created to repeatedly execute the “ping” command.

An example of an embedded script can be found here:

C:\Program Files (x86)\Keysight\RFSensor\Embedded Apps Examples\KeepAlivePinger

The sensor uses BusyBox (v1.4.2) for its "sh" shell and built-in commands. Here's a list of busybox commands available on the sensor:

[, [[, addgroup, adduser, arp, arping, ash, awk, basename, bunzip2, bzcat, cat, chgrp, chmod, chown, chroot, clear, cmp, cp, cpio, crond, crontab, cut, date, dc, dd, delgroup, deluser, df, diff, dirname, dmesg, dos2unix, du, echo, egrep, env, expr, false, fgrep, find, free, ftpget, ftpput, fuser, getty, grep, gunzip, gzip, halt, head, hexdump, hostid, hostname, id, ifconfig, inetd, init, insmod, install, ipcalc, ipcs, kill, killall, klogd, less, ln, logger, login, logread, ls, lsmod, md5sum, mkdir, mknod, mktemp, modprobe, more, mount, mv, nc, netstat, nice, nmeter, nohup, nslookup, od, passwd, pidof, ping, pivot_root, poweroff, printenv, printf, pwd, readlink, reboot, reset, resize, rm, rmdir, rmmod, route, sed, sh, sleep, sort, start-stop-daemon, stat, strings, stty, su, sulogin, switch_root, sync, syslogd, tail, tar, tee, telnet, test, tftp, time, top, touch, tr, traceroute, true, tty, udhcpc, umount, uname, uniq, unix2dos, unzip, uptime, usleep, vi, watchdog, wc, wget, which, whoami, xargs, yes, zcat, zcip

Create the tar.gz installer file

Before installing an embedded app or script on a sensor, it has to be packaged it into a .tar.gz installer file first. The following steps outline this process:

Application name rules:

The app name must NOT contain an underscore _.

Spaces in the app name are not recommended. They may introduce unexpected bugs.

The app name is case sensitive. All files using the app name must use the same case. The same app name is used in the SAL embedded app APIs, such as salAppGetAppSettings() and salAppSetAppSettings().

Examples: KeepAlivePinger, GPSLocationLogger.

Installer file name

The .tar.gz installer file must be <appname>_<version info>.tar.gz.

The sensor firmware ignores the version info string, so the version info string may be formatted in any the app requires. Examples: "1.0", "1.0.0", "Ver.A". The embedded app can use the version info string to keep track of which version of the embedded app or script is currently installed on a sensor.

Examples of a .tar.gz installer file can be found here:

C:\Program Files (x86)\Keysight\RFSensor\Embedded Apps Examples\KeepAlivePinger\KeepAlivePinger_1.0.0.tar.gz

C:\Program Files (x86)\Keysight\RFSensor\Embedded Apps Examples\GPSLocationLogger\GPSLocationLogger_1.0.0.tar.gz

Installer files

The .tar.gz installer file must contain these files in addition to the executable or shell script:

configure<appname>.sh

The sensor firmware calls this shell script to start, stop, restart, uninstall, or perform other operations on the embedded app.

This configure shell script must support the following command line options:

-start
-stop
-restart
-uninstall
-status
-version

Examples of a configure script can be found here:

C:\Program Files (x86)\Keysight\RFSensor\Embedded Apps Examples\KeepAlivePinger\configureKeepAlivePinger.sh

C:\Program Files (x86)\Keysight\RFSensor\Embedded Apps Examples\GPSLocationLogger\configureGPSLocationLogger.sh

<appname>.ini

This file contains settings for an embedded app. The embedded app can use the SAL APIs salAppGetAppSettings() and salAppSetAppSettings() to read/write the settings in this file.

The format of the key/value pairs is <key>=<value>.

The settings file must contain the "autoStart" key. If autoStart is a nonzero value, an embedded app or script is started when the sensor boots. Example: autoStart="1"

Examples of a .ini settings file can be found here:

C:\Program Files (x86)\Keysight\RFSensor\Embedded Apps Examples\KeepAlivePinger\KeepAlivePinger.ini

C:\Program Files (x86)\Keysight\RFSensor\Embedded Apps Examples\GPSLocationLogger\GPSLocationLogger.ini

install.sh

The sensor firmware calls this shell script to decompress the .tar.gz file and copy the files to the proper directories. Rules:

- install.sh must copy "configure<appname>.sh" to "/jffs2/apps".
- install.sh must copy "<appname>.ini" to "/jffs2/apps/registry".
- install.sh must copy app's binaries or scripts into "/jffs2/apps".

Examples of an install.sh shell script can be found here:

C:\Program Files (x86)\Keysight\RFSensor\Embedded Apps Examples\KeepAlivePinger\install.sh

C:\Program Files (x86)\Keysight\RFSensor\Embedded Apps Examples\GPSLocationLogger\install.sh

Embedded app executable binaries, scripts, and other Linux libraries.

SAL library files "libsal.so" and "libdmc-linux.so" do not need to be included in the installer file. Those 2 files are already in the sensor's Linux file system.

Install/uninstall/ run an embedded App

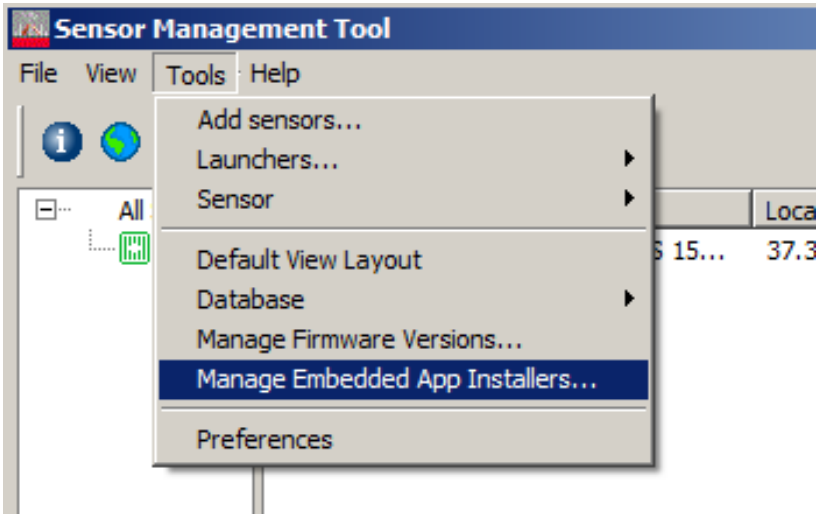
An embedded applications' .tar.gz file can be installed onto the sensor in one of the two ways:

– Add it to the Sensor Management Server (SMS) and then install it on each sensor

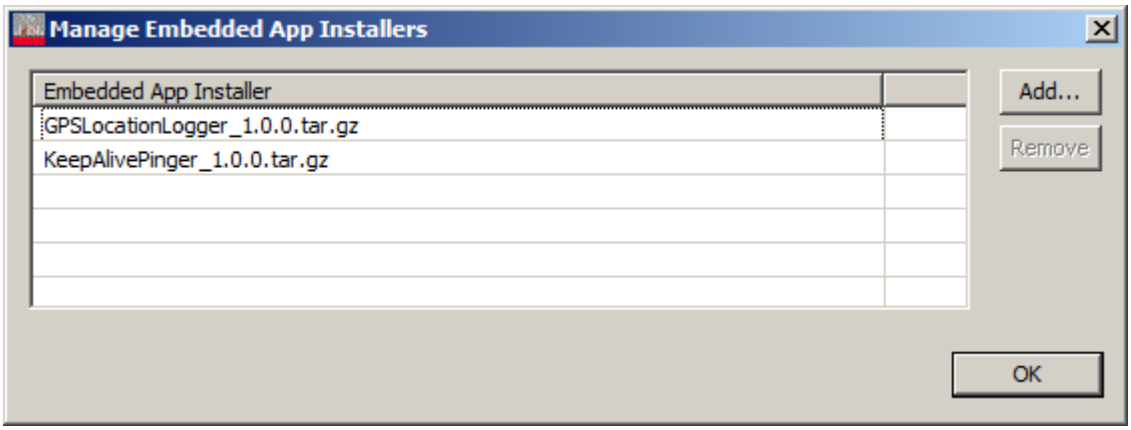
– Directly install/uninstall/run it on each sensor through the SAL API.

  See agSalEmbeddedApp.h: salAppInstallApp(), salAppUninstallApp(), salAppStartApp(), etc.

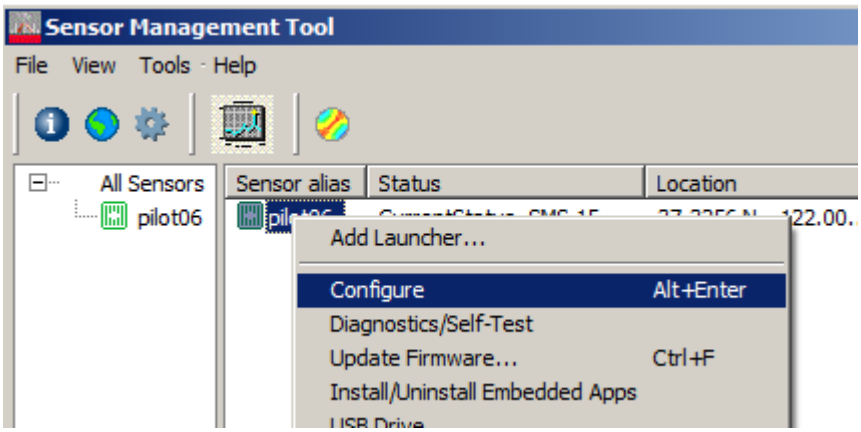
To add.tar.gz installer file to the SMS, launch the Sensor Management Tool (SMT), go to Tools -> Manage Embedded App Installers.



A dialog will open where tar.gz installer files can be added to or removed from the SMS.

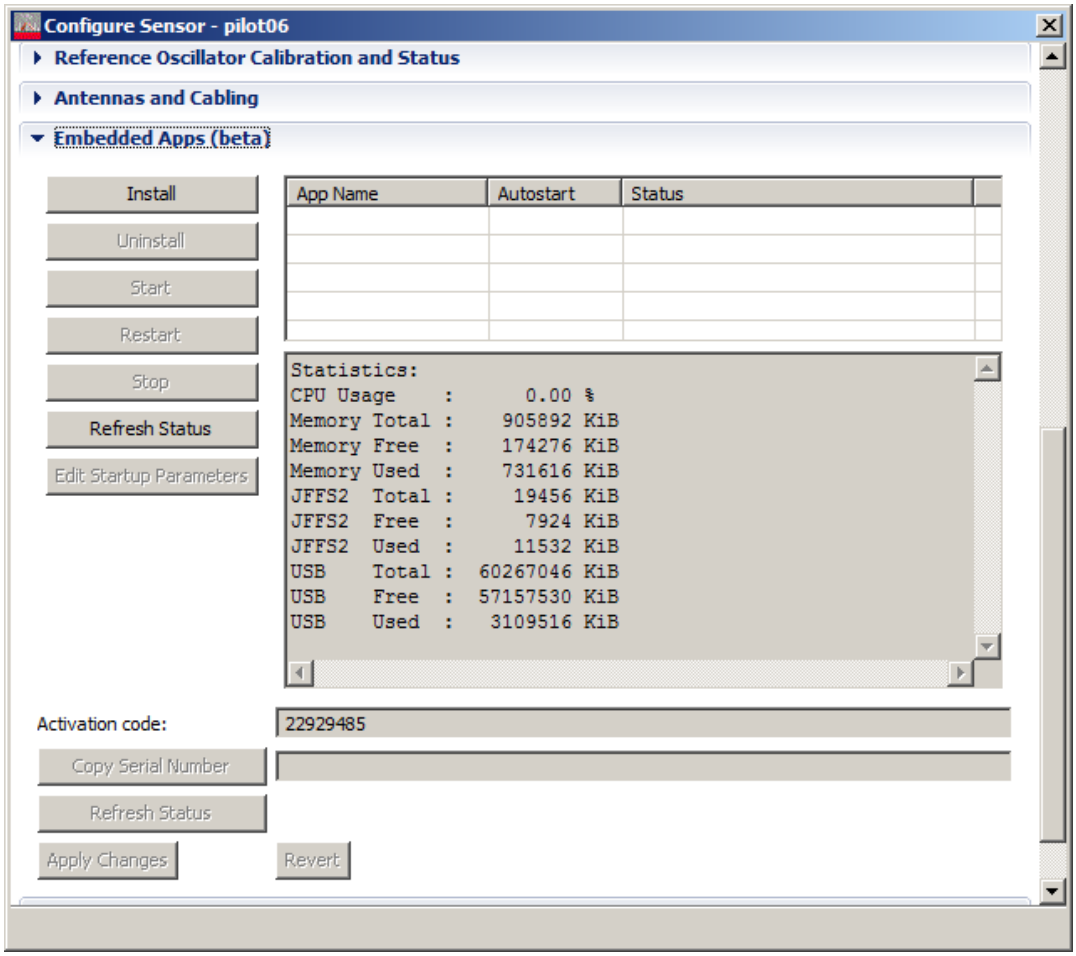


To install an embedded app on a specific sensor, right click on a sensor -> Configure. This will launch the Configure Sensor window.



Click on the "Embedded Apps" section, then click the Install button, and choose the embedded app to install onto the sensor.

It is also possible to uninstall, start, restart, stop, or get the status of the embedded app here.



Debug

Without access to the sensor's Linux console, some alternative ways might aid debugging of embedded applications:

– Log file - residing on a USB drive, written to by the embedded application. That file can be downloaded via the Sensor Management Tool GUI to examine errors/status information.

– Network server socket - On the PC side, a telnet client session to this socket (on a specified port number) can allow commands to be sent to the embedded application and to receive output from it.

– Check the CPU usage – read the contents of /proc/loadavg. The first three fields measure CPU and IO utilization of the last 1, 5, and 10-minute periods. The fourth field shows the number of currently running processes and the total number of processes. The last field displays the last process ID used.

– salSensorBeep() - can be used to determine which code path is executing or give audible clues to errors encountered.

Edition 1.3, April 2018

N6841-90005

www.keysight.com