LANTRONIX®

Application Note:

Using the X300 Series SDK

Part Number APP-0124
Revision B May 2023

Intellectual Property

Contacts

Lantronix, Inc.
48 Discovery, Suite 250
Irvine, CA 92618, USA
Toll Free: 800-526-8766
Phone: 949-453-3990
Fax: 949-453-3995

Technical Support
Online: www.lantronix.com/support

Sales Offices
For a current list of our domestic and international sales offices, go to the Lantronix web site at www.lantronix.com/about/contact

Disclaimer

All information contained herein is provided “AS IS.” Lantronix undertakes no obligation to update the information in this publication. Lantronix does not make, and specifically disclaims, all warranties of any kind (express, implied or otherwise) regarding title, non-infringement, fitness, quality, accuracy, completeness, usefulness, suitability or performance of the information provided herein. Lantronix shall have no liability whatsoever to any user for any damages, losses and causes of action (whether in contract or in tort or otherwise) in connection with the user’s access or usage of any of the information or content contained herein. The information and specifications contained in this document are subject to change without notice.

Revision History

Date	Rev.	Comments
July 2022	A	Initial document.
May 2023	B	Added Secure Boot.

For the latest revision of this product document, please check our online documentation at www.lantronix.com/support/documentation.

Contents

Using the X300 Series SDK

This document describes how to use the Lantronix X300 Series SDK to create custom packages and Lantronix Image Builder to build custom firmware images. It also describes how to customize UI items using Image Builder.

This document assumes that you have a computer running a supported Linux distribution (or VM) and are familiar with the OpenWrt build system and the Lantronix Web Admin for configuring X300 series devices.

The command examples use the Bash command interpreter in a development environment.

Additional Documentation

The following documentation resources provide background and reference information.

- OpenWrt online documentation (refer to developer guide)
- *Lantronix X300 Series IoT Cellular Gateway User Guide*

Download the SDK Files

To download the Lantronix X300 Series SDK, contact Lantronix Technical Support.

The SDK includes the following components:

- Lantronix X300 Series SDK
- Lantronix Image Builder

The Lantronix X300 Series SDK is a pre-compiled environment suitable for creating custom packages to install on the device or build into the firmware. It's provided as a tar.xz archive.

The Lantronix Image Builder is a pre-compiled environment suitable for creating custom images for Lantronix X300 series devices. It's also provided as a tar.xz archive.

OS Requirements and Prerequisites

The SDK and Image Builder run in 64-bit Linux. It is recommended that you use a Linux environment, standalone or virtual machine.

The prerequisites are the same as for the OpenWrt build system. Please refer to the OpenWrt Developer guide for build system setup, build prerequisites in different Linux distributions, and package prerequisites.

Creating a Custom Package Using the SDK

Use the SDK to create a custom package. The custom package can then be included into a firmware image using Image Builder, or alternatively, it can be installed using the OPKG package manager (see the *Lantronix X300 Series IoT Cellular Gateway User Guide*).

Sample Application

You will need a sample application to create the custom package. If you don't already have an application, you can create a simple "Hello, world" application with a structure similar to the following:

```
<packagename>/
  |
  +-- Makefile
  +-- src/
       |
       +-- source and include files
       +-- Makefile
```

Note: *<packagename> must be one word with no spaces.*

Extract the SDK

To extract the SDK archive use the following command.

```
tar -xvf <sdk-filename.tar.xz>
```

replacing `sdk-filename` with the actual file name of the tar archive.

Add Your Package Feeds

The `feeds.conf.default` file in the directory where you extracted the SDK contains the list of package feeds. You should modify this file in the following cases:

- If the package requires additional dependency packages supplied by OpenWrt, add the repository, or change to the proper repository.
- If the package requires additional dependency packages supplied by your own (custom) repository, add your own repository.
- See https://openwrt.org/docs/guide-developer/feeds

Make the Custom Package

Note: *All Linux commands should be run as user. Do not use root or sudo.*

1. In the `package/` directory, create a directory for the package. The name of the directory should not include any spaces. For this example we'll call it "packagename".
2. Navigate to the "packagename" directory, and create or copy a Makefile.
3. In the "packagename" directory, copy or create the `src/` directory.

4. Navigate to the `src/` directory and copy or create the Makefile mentioning how to build the package. This can be ignored and can also be done from `package/Makefile` in the parent directory. (Refer to Openwrt online documentation for Makefile details.)
5. Still in the `src/` directory, copy or create the source files and include files.
6. If the package requires additional dependency packages having no supplier/repository, then you will need to create those packages as described in steps 1-5.
7. If the package requires additional dependency packages supplied by OpenWrt or by your own repository, modify the `feeds.conf.default` file as described in the previous section, *Add Your Package Feeds*. If you add or modify feeds, use the following commands to update and install the feeds:

   ```
   ./scripts/feeds/update -a
   ./scripts/feeds/install -a
   ```

8. Run `make menuconfig` to open the SDK's menu and select the package. Find the package that you want to build and select it by pressing "m". This will also select all the dependencies. Save the configuration and exit the menu.
9. To compile the package, run:

   ```
   make package/<packagename>/compile
   ```

   Note: *You can use* `V=s` *option for detailed logs of the build.*

10. After the package is compiled, the generated .ipk file (or files) are created in the `bin/target/at91/ltrx_sam9x60/packages/` directory.

Integrate the Custom Package into an Image

Complete these steps to integrate the package you just compiled into the image. The image build steps are described in detail in the next section, *Building an Image using Image Builder*.

1. Copy the .ipk (or .ipk files) that you created in the previous procedure and paste it into the `packages/` directory of the Image Builder.
2. From the SDK directory, read the `tmp/.packageinfo` file to find the package and dependencies information. Refer to the following example for the "helloworld" package:

```
**************************************************************************
Source-Makefile: package/helloworld/Makefile

Package: helloworld
Submenu: Bin Packages
Version: 0-1
Depends: +libc +GCC_LIBSSP:libssp +USE_GLIBC:librt +USE_GLIBC:libpthread
Conflicts:
Menu-Depends:
Provides:
```

```
Section: utils
Category: Lantronix
Repository: base
Title: HelloWorld
Maintainer:
Source:
Type: ipkg
Description:  This is a sample HelloWorld program for SDK.

@@

**************************************************************************
```

3. Copy all of the information for the package and its dependencies as shown above into `.packageinfo` of the Image Builder. Be careful to ensure that the information is correct before proceeding because if this fails then the build will fail.
4. Follow the procedure for building the image as described in the next section. Put the package names and dependencies in the <addCustomPackages> field of the PACKAGES variable.

Building an Image using Image Builder

The Lantronix Image Builder is a pre-compiled environment suitable for creating custom images for Lantronix X300 series devices.

Extract Image Builder

To extract the Image Builder use the following command.

```
$ tar -xvf <imagebuilder-filename.tar.xz>
```

replacing `imagebuilder-filename` with the actual file name of the tar archive.

Usage

To build the image, use the `make image` command. To customize the image, make use of the following variables:

- `PROFILE`
- `PACKAGES`
- `FILES`
- `DISABLED_SERVICES`

Run `make help` to get detailed help information.

Select the Profile

The PROFILE variable specifies the target image to build.

```
PROFILE="profile-name"
```

For the X300 image, the profile name should be "hill".

Run `make info` to get the list of available profiles.

Select Packages

The PACKAGES variable lets you include or exclude packages in the firmware image.

```
PACKAGES="pkg1 pkg2 pkg3 -pkg4 -pkg5 -pkg6"
```

Package names are space separated. Excluded packages are preceded by a minus sign "-".

In the usage example above, pkg1, pkg2, and pkg3 are included in the firmware image and pkg4, pkg5, and pkg6 are excluded from the firmware image.

Included Packages

To get the list of installed packages and IPK packages:

1. To get the list of installed packages, enter the following command.

   ```
   $ cat .config | grep -v '=m' | grep -v 'is not set'| grep CONFIG |
   awk -F'^CONFIG_' '{print$2}' | sed -E 's/PACKAGE_//' | grep -v
   '^DEFAULT' | awk -F'=' '{print$1}' > installed_packages.txt
   ```

2. To get the list of IPK packages, enter the following command.

   ```
   $ cat .config | grep -v '=y' | grep -v 'is not set'| grep CONFIG |
   awk -F'^CONFIG_' '{print$2}' | sed -E 's/PACKAGE_//' | grep -v
   '^DEFAULT' | awk -F'=' '{print$1}' > ipk_packages.txt
   ```

3. To separate the above new line values into space separated values and remove all CAPITAL strings, use the following command.

   ```
   $ variable=`cat <installed_packages.txt or ipk_packages.txt> | grep
   -v '[!A-Z]'`
   ```

   Note: *Run this command once for installed_packages.txt and once for ipk_packages.txt.*

List the <InstalledPackages> and <IPK Packages> in the PACKAGE variable in the `make` command.

Excluded Packages

The following packages will be excluded when the image is built.

> Ignored Packages == -arm -odhcp6c_ext_cer_id -odhcpd-ipv6only -odhcpd_ipv6only_ext_cer_id -luci-lib-nixio_notls

List the ignored packages in the PACKAGE variable in the `make` command.

Any other packages to be excluded should be listed in the PACKAGES variable, preceded by a minus sign "-".

Note: *If the excluded package is a dependency for another package, the build may fail.*

Including Custom Packages

To include custom packages with the image, follow the instructions in *Creating a Custom Package Using the SDK* to compile and integrate the package files (IPKs) and then list them in the PACKAGES variable.

Add Custom Files

The FILES variable lets you include custom configuration files into the firmware as default values.

```
FILES= "files"
```

To add custom files, create a "files" directory in the Image Builder root directory where you issue the make command. Create a source tree that matches the source tree where the files need to be copied or replaced in the device.

Use the FILES variable in the `make` command.

Disable Services

The DISABLED_SERVICES variable lists the names of services from /etc/init.d to be disabled. Multiple services should be space separated.

```
DISABLED_SERVICES="svc1"
```

where "svc1" is the name of a service from /etc/init.d

Use the DISABLED_SERVICES variable in the `make` command.

Make the Image

Note: *Use* `make help` *command to show help information.*

1. To build the image, use the `make image` command.

   ```
   make image PROFILE="<Profile name>" PACKAGES="<InstalledPackages> +
   <Installed IPKs> + <IgnoredPackages>"
   ```

2. To customize the image build:
   - To add custom packages, list them in the PACKAGES variable.
   - To exclude packages, list them in the PACKAGES variable (preceded by a minus sign "-".)
   - To add custom files into the firmware as default values, append the FILES variable.
   - To disable inbuilt services from init.d, append the DISABLED_SERVICES variable.

   ```
   make image /
   PROFILE="<Profile name>" /
   PACKAGES="<InstalledPackages> + <Installed IPKs> + <IgnoredPackages>
   + <customPackages> + -<unwantedPackages>" /
   ```

```
FILES="files" /
DISABLED_SERVICES="<nameOfServices>"
```

(If copying this example, remove the forward slashes.)

3. After completion of the image build, the build files will be output to the `bin/targets/at91/ltrx_sam9x60/` directory of the Image Builder.
4. To list packages that get installed into the image. (Use the `make help` command to get more details). Two examples are provided below:

```
make manifest PROFILE="<Profile name>"
```

//To list all packages that get installed in default firmware

```
make manifest PROFILE="<Profile name>"
PACKAGES="<addCustomPackages>"
```

//To list all packages that get installed in firmware after additional custom packages

UCI Defaults

OpenWrt relies on UCI, the Unified Configuration Interface, to configure its core services. UCI defaults provides a way to preconfigure your images, using UCI.

To set some system defaults when the device is built, it is possible to create a script in the `<imagebuilder>/files/etc/uci-defaults/` directory. Scripts in that folder are copied to the device and executed by `/etc/init.d/boot` the first time the device boots. If the scripts exit with code 0 they are deleted afterwards. Scripts that exit with non-zero exit code are not deleted and will be re-executed at the next boot until they also successfully exit.

To update the runtime configuration using a uci-defaults script:

1. Under the `<imagebuilder>/files/etc/` directory, create a folder called "uci-defaults".
2. Under the "uci-defaults" folder, create a script file called "99-default-settings".
3. Use SSH to access your X300 device and check for the parameters that you want to change. Use the `uci show <config>` command to get the exact parameter name.

   Example (show configuration parameters from network file):

   ```
   uci show network
   ```

   Example output:

   ```
   network.cellular.simselect='2'
   network.wanlan=wanlan
   network.wanlan.enable='0'
   network.wanlan.done='0'
   network.watchdog=watchdog
   ```

4. In the "99_default_settings" script file, create a script like the example shown below: Use `uci set` (line 3) command to set the value of the given option and `uci commit` (line 4) to write the changes to the filesystem. For multiple parameters, enter each `uci set` command on a new line before the `uci commit` command. Lines 1, 6, and 7 are fixed.

5. Give execute permission to "99_default_settings" script using the following command:

   `chmod +x 99_default_settings`

6. Follow the Image Builder procedure to make the image (use the FILES variable in the make command) and install the firmware on the device.

Device Configuration

There are two ways to update the device configuration. If you will modify only a few parameters or if the parameter should be modified after the image is built, use the method as described in the previous section, *UCI Defaults*. Otherwise, if there are more than a few changes, use the procedure described below.

To update configuration:

1. Prepare the device with the custom configurations.
2. Copy the modified files from the `/etc/config` directory from the device.
3. Paste the copied files under the `<imagebuilder>/files/etc/config/` folder.
4. Follow the Image Builder procedure to make the image (use the FILES variable in the make command) and install the firmware on the device.

Note: *Device specific parameters in the copied config folder might also get changed. To undo those changes, use the "uci-defaults" script.*

Customizing the UI Using Image Builder

Use the Image Builder to customize the UI and other modifications.

Firmware File Name and Configuration Version

Change the firmware file name and the configuration version by modifying the `.config` file in the <imagebuilder> directory.

To change the firmware file and configuration version:

1. Change the following options in the `.config` file:
   - CONFIG_VERSION_DIST="<Lantronix>"
   - CONFIG_VERSION_NUMBER="<version-number>"

   Important: *Do not change any other parameters in the .config file, as doing so may cause the image build to fail.*
2. To view that the file name has been modified, build the image and check the output in the `<imagebuilder>/bin/targets/at91/ltrx_sam9x60/` directory.

Customize UI Elements

Image Builder provides ways to modify the current theme (logo, icons, stylesheet) and customize other items such as the SSH login banner and certain text strings and parameters.

Note: *The SDK does not support replacing the UI theme.*

To customize UI elements:

1. View the directory structure of the files on the device by navigating to the following directory in Image Builder and drilling down to find the desired UI theme file and other parameter.

   ```
   ROOT="<imagebuilder>/build_dir/target-arm_arm926ej-
   s_musl_eabi/root.orig-at91/"
   ```
2. Go to the `<imagebuilder>/files` directory (or create "files" directory if it doesn't exist) and create a source tree that matches the source tree where the files need to be copied or replaced.
3. Copy the file from the ROOT and paste it under the "files" directory. Modify or replace the file according to your requirement.
4. Build and install the image to view that the UI related files have been replaced in the device.

Refer to *Table 1* for a list of commonly customized elements. The numbers in the table correspond to the labels in *Figures 1-4* showing the location of the elements on the user interface.

Label	UI Element	File Location/Description
1	SSH banner	"ROOT/etc/banner"
2	Login page logo	"ROOT/www/luci-static/rosy/logo.png"
3	Logo on all other pages in Web Admin	"ROOT/www/luci-static/rosy/logo_t.png"
4	Other Icons/logos in Web Admin	"ROOT/www/luci-static"
5	Default hostname	See *UCI Defaults* or *Device Configuration*.
6	"Powered by" string	"ROOT/etc/openwrt_release" Modify DISTRIB_DESCRIPTION parameter.
7	Quick Setup information	"ROOT/usr/lib/lua/luci/view/quicksetup/quicksetup.htm"
8	Stylesheet – fonts, sizes, color, styles	"ROOT/www/luci-static/rosy/cascade.css"
9	Default AP SSID	See *UCI Defaults* or *Device Configuration*.

Table 1: UI elements

Example:

To customize the SSH login banner as shown in *Figure 1*, copy the banner file from the `ROOT/etc/` directory. Navigate to the `<imagebuilder>/files` directory and create the `etc/` directory. Paste the banner file here and modify it as per your requirement. Build the image using Image Builder and install the firmware on the device.

Figure 1: SSH Login Banner

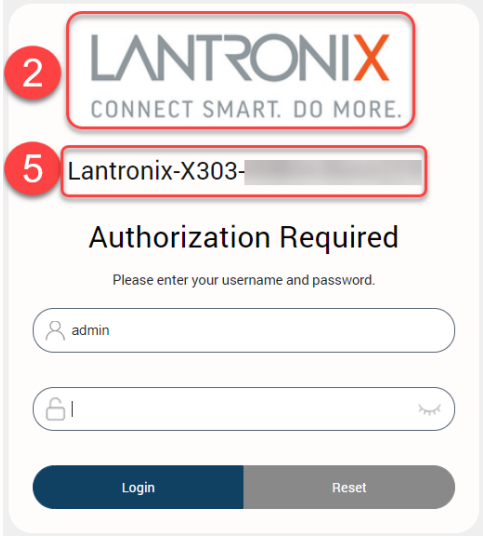

Figure 2: Web Administration Login page

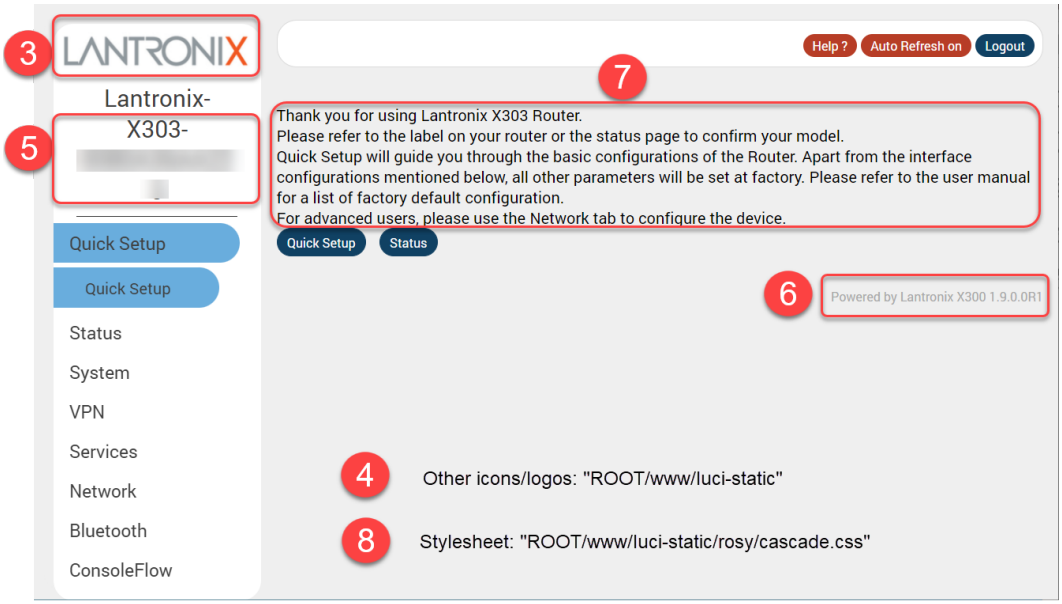

Figure 3: Quick Setup page (logged in user)

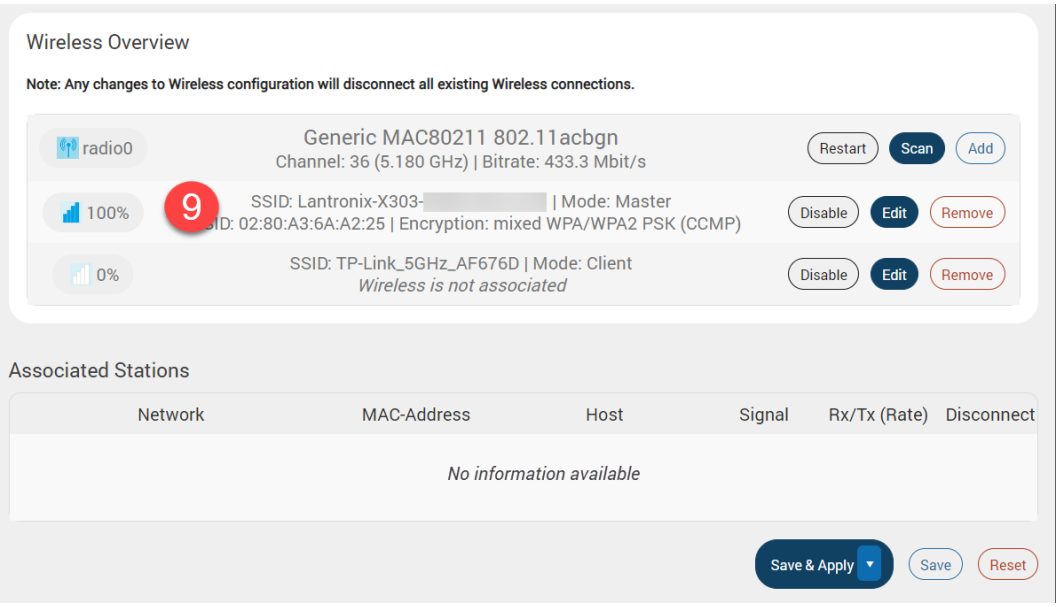

Figure 4: Default AP SSID

Secure Boot

Secure Boot ensures that only digitally signed software is run on the X300. If you plan to release custom firmware, you must prepare the X300 for OEM Secure Boot using the process below.

Preparing the X300 to use custom firmware requires the use of the serial port on the X300. X300 models that do not have a serial port are unable to use custom firmware.

Firmware Filenames

The following files are used in the process of preparing the X300 for Secure Boot. The version number may be different.

- X300_<version>.rom - Application firmware
- sam9x60_mfgtestldr_<version>.rom - MFG loader
- sam9x60_recovldr_<version>.rom - Recovery loader

Preparing the X300 for OEM Secure Boot

1. Create an OEM public-private key pair using the following command:

```
openssl ecparam -name secp256r1 -genkey -out oem-priv.pem
openssl ec -in oem-priv.pem -pubout -out oem-pub.pem
```

2. Use the request form to submit the public key to Lantronix for signature. Lantronix returns the signed public key as a .rom file that is named similarly to optional_rsa_key_pub.signed.rom.
3. Sign the application firmware with the OEM private key.

```
ltrx-signimage -f oem-priv.pem X300_<version>.signed.rom
X300_<version>.oem.signed.rom
```

4. Launch the MFG loader (sam9x60_mfgtestldr_<version>.signed.rom) by connecting to the device over serial using a terminal emulator such as TeraTerm and sending the following command until the G prompt appears:

```
!SL
```

5. Send the MFG loader file (sam9x60_mfgtestldr_<version>.signed.rom). In TeraTerm, this is done by clicking **File > Send File**.
6. Install the Lantronix-signed OEM key (oem-pub.signed.rom).

```
flash download serial
```

7. Lock the OEM key.crypto lock-key oem-key
8. Download the OEM-signed application firmware (X300_<version>.oem.signed.rom).

```
flash download serial
```

9. Reboot.

Note: *You will need to use the OEM-signed MFG loader, recovery loader, and application firmware once the OEM key has been configured on the device.*

10. Sign the MFG test loader with the OEM private key.

```
ltrx-signimage -f oem-priv.pem sam9x60_mfgtestldr_<version>.signed.rom
sam9x60_mfgtestldr_<version>.oem.signed.rom
```

11. Sign the Recovery Loader with the OEM private key.

```
ltrx-signimage -f oem-priv.pem sam9x60_recovldr_<version>.signed.rom
sam9x60_recovldr_<version>.oem.signed.rom
```

Releasing Custom Firmware

Once Secure Boot is enabled on the gateway, all custom firmware must be signed with the OEM private key using the ltrx-signimage application. This is done each time you release a firmware rom.

```
ltrx-signimage -f ltrx-priv.pem X300_<version>.signed.rom
X300_<version>.oem.signed.rom
```

After following the procedure above to prepare the X300 for OEM Secure Boot, no additional steps with the device are required when releasing new firmware.