

Application Note:

Using the ePack SDK

Intellectual Property

© 2021 Lantronix, Inc. All rights reserved. No part of the contents of this publication may be transmitted or reproduced in any form or by any means without the written permission of Lantronix.

Lantronix is a registered trademark of Lantronix, Inc. in the United States and other countries.

Patented: <http://patents.lantronix.com>; additional patents pending.

All trademarks and trade names are the property of their respective holders.

Contacts

Lantronix, Inc.

7535 Irvine Center Drive, Suite 100

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
March 2021	A	Initial document.

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

Using the ePack SDK

This document describes how to use the Lantronix ePack 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 E21x and E22x 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 ePack User Manual for E210 and E220 Series Devices](#)

Download the SDK Files

To download the Lantronix ePack SDK, contact [Lantronix Tech Support](#).

The SDK includes the following components:

- Lantronix ePack SDK
- Lantronix Image Builder

The Lantronix ePack 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 E21x or E22x 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 ePack User Manual*).

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 directory 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.

To 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 [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/ramips/mt7620/packages/` directory.

Integrating 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 E21x or E22x 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.

Make Image Using Image Builder

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

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`.

- The following packages will be ignored. Note the minus sign "-" before the package name of excluded packages.

Ignored packages ==

```
-mipsel -odhcp6c_ext_cer_id -odhcpd-ipv6only -odhcpd-ipv6only_ext_cer_id -luci-lib-nixio_notls
```

- To build the image, use the following command. (Do step 5 or 6 or 7, but not all three steps.)

```
$ make image PROFILE="<Profile name>" PACKAGES="<Step_3_output> + <IgnoredPackages>"
```

To add custom packages or exclude unwanted packages use the following command.

```
$ make image PROFILE="<Profile name>" PACKAGES="<Step_3_output> + <IgnoredPackages> + <addCustomPackages> + -<unwantedPackages>"
```

- PROFILE - The profile name should be "e21x" or "e22x"
 - PACKAGES – **Note:** Package names are space separated. Excluded packages are preceded by a minus sign "-".
 - `<Step_3_output>` - This is the list of installed and IPK packages in the image.
 - `<IgnoredPackages>` - For ignored packages, copy the list from step 4.
 - `<addCustomPackages>` - To add custom packages, follow the instructions in [Creating a Custom Package using the Lantronix SDK](#) to compile and integrate the package files (IPKs) and then list them here.
 - `<unwantedPackages>` - List the unwanted packages with a "-" before each package. Note that if the unwanted package is a dependency for another package, the build may fail.
 - Output - After completion of the image build, the build files will be output to the `bin/targets/ramips/mt7620/` directory of the Image Builder.
- To add any custom files into the firmware as default values, create a "files" directory in the Image Builder directory and create a source tree that matches the source tree where the files need to be copied or replaced in the device.

Then use the FILES variable in the make command.

```
$ make image PROFILE="<Profile name>" PACKAGES="<Step_3_output> + <IgnoredPackages> + <addCustomPackages> + -<unwantedPackages>" FILES="files/"
```

After completion of the image build, the build files will be output to the `bin/targets/ramips/mt7620/` directory of the Image Builder.

7. To disable inbuilt services from `/etc/init.d/`, use the `DISABLED_SERVICES` variable in the `make` command.

```
$ make image PROFILE="<Profile name>" PACKAGES="<Step_3_output> +  
<IgnoredPackages> + <addCustomPackages> + -<unwantedPackages>"  
FILES="files/" DISABLED_SERVICES="<nameOfServicesSpaceSeparated>"
```

After completion of the image build, the build files will be output to the `bin/targets/ramips/mt7620/` directory of the Image Builder.

8. 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="e22x" //To list all packages that get installed  
in default firmware
```

```
make manifest PROFILE="e22x" PACKAGES="<addCustomPackages>"  
//To list all packages that get installed  
in firmware after additional custom  
packages
```

Customize the UI Using Image Builder

You can 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/ramips/mt7620/` 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. This section describes how to customize UI elements and their location in the device's source tree (see Table 1).

Note: The SDK does not support replacing the UI theme in the current release.

To customize UI components:

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-mipsel_24kc_musl/root.orig-ramips/"
```

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.

For example, to customize the SSH 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.

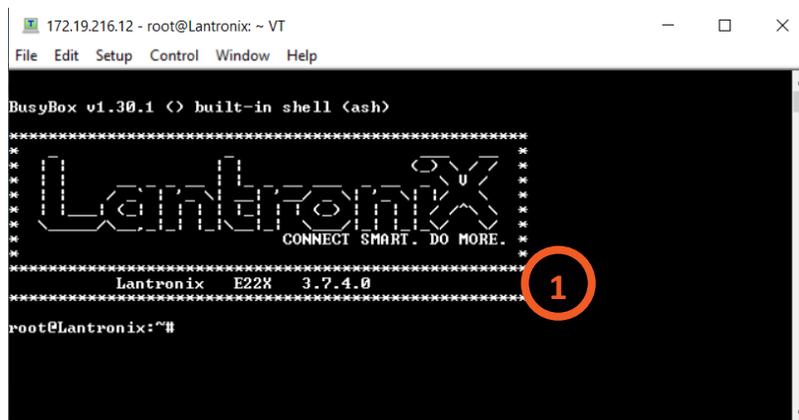


Figure 1: SSH Login Banner

Table 1 shows customizable UI components and their location in the source tree. The numbers in the table correspond to the labels for Figures 1, 2, and 3 showing the location of the components in the user interface.

Label	UI component	File or folder location
1	SSH banner	"ROOT/etc/banner"

Label	UI component	File or folder location
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	"ROOT/bin/config_generate" Modify "set system.@system[-1].hostname=<HOSTNAME HERE>" parameter
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"

Table 1: UI components and their location

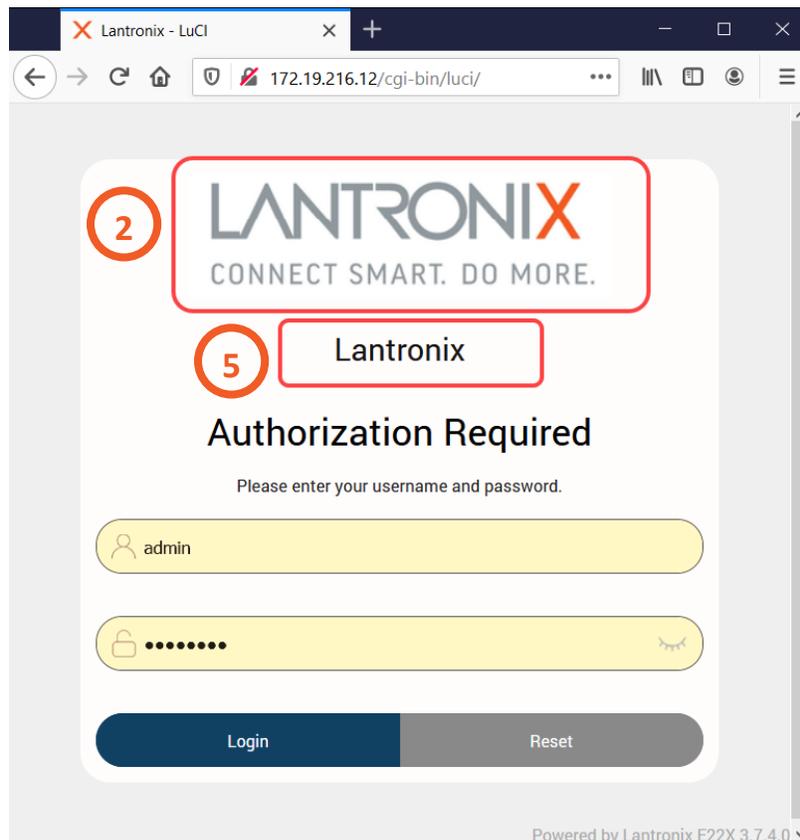


Figure 2: ePack Login page

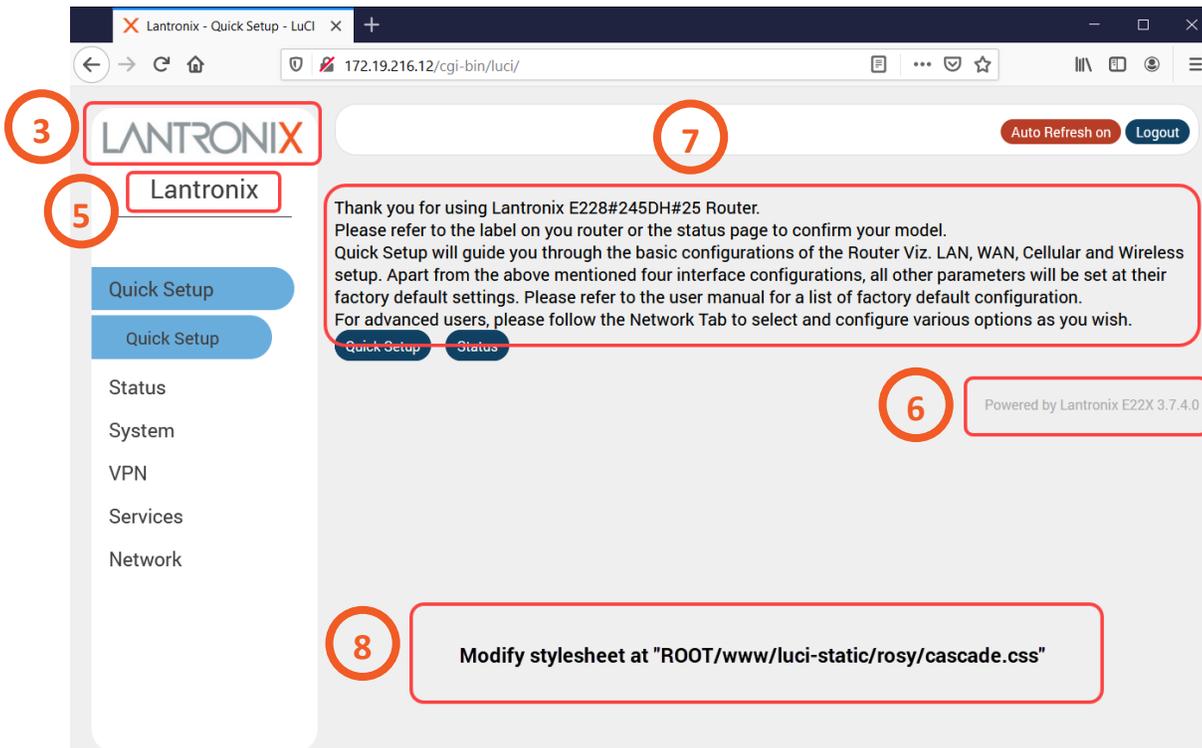


Figure 3: ePack Quick Setup page (logged in user)