Build RPMs for Native Application Hosting

This section explains how you can build RPMs for native application hosting.

Set Up the Build Environment

This section describes the two methods of preparing and setting the build environment for native application hosting.

Create Native Build Environment Using QEMU Hypervisor

This section describes a method of creating the native Wind River Linux 7.0 build environment, and running the environment ISO by using a Quick Emulator (QEMU) hypervisor.

Prerequisites

- Ensure that you have access to the Cisco repository containing the native WRL7 ISO.
- Download the native ISO with the .iso extension.

Configuration Procedure

1 Launch the native WRL7 ISO and install it onto a disk image.

```
qemu-system-x86_64 -m 16G -cdrom <path-to-the-downloaded-iso-file> -net nic -net user
-hda ./wrl7.img
 -cpu core2duo -show-cursor -usb -usbdevice wacom-tablet -vga vmware
```

2 Relaunch the native build environment with the installed image.

```
qemu-system-x86_64 -m 16G -net nic -net user -hda ./wrl7.img -cpu core2duo -show-cursor
 -usb -usbdevice wacom-tablet -vga vmware
```

The native build environment is ready for hosting third-party applications. The user is connected to the VGA console port of the native QEMU VM.

Alternatively, a user can connect to an SSH service running inside the VM.

Create a Cross-Build Environment Using the SDK Shell Script

As an alternative to the native environment, you can create a cross-build environment by using a WRL7 cross-SDK shell script. You can install the SDK by running the shell script on a general-purpose Linux environment, such as an Ubuntu 14.04 host machine.

Prerequisites

Ensure that the following requirements are met before you proceed with the installation.

- Access to the SDK in the Cisco repository.
- Ability to build any customization, as needed, on the SDK.

Installation Procedure

To install the SDK for native application hosting, use these steps:

1. Download the SDK from the Cisco repository.

```
wget https://devhub.cisco.com/artifactory/xr600/app-dev-sdk/x86_64/
wrlinux-7.0.0.2-glibc-x86_64-intel_x86_64-wrlinux-image-glibc-std-sdk.sh
```

2. Install the SDK by running the shell script.

```
john@sjc-ads-4587:john$
./wrlinux-7.0.0.2-glibc-x86_64-intel_x86_64-wrlinux-image-glibc-std-sdk.sh
```

3. Enter the target directory for installing the SDK.

   Choose a target directory that has sufficient storage space.

```
Enter target directory for SDK
(default: /opt/windriver/wrlinux/7.0-intel-x86-64):
/nobackup/john/sdk_extract
You are about to install the SDK to "/nobackup/john/sdk_extract". Proceed[Y/n]? Y
```

   On successful installation, a message is displayed on the screen.

```
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
```

The SDK for native application hosting is successfully installed.

What to do Next

You can set up the environment variables, and validate them as explained in this section.

1. Navigate to the directory, where the SDK is installed and set up the environment variables by running the following commands:
   - If you are using a bash shell, run the **. ./env.sh** command.
   - For any other shell, run the **source . ./env.sh** command.

   The commands execute the environment setup file that was extracted during SDK installation.
2. Validate the installed environment variables by running the **env** command to view all variable values.

3 Validate the CC environment variable by running the **env | grep CC** command, and verifying whether the following value is assigned:

```
CC=x86_64-wrs-linux-gcc  -m64
--sysroot=/nobackup/john/sdk_extract/sysroots/core2-64-wrs-linux
```

Alternatively, you can use the echo command:

```
echo $CC
x86_64-wrs-linux-gcc -m64
--sysroot=/opt/windriver/wrlinux/7.0-intel-x86-64/sysroots/core2-64-wrs-linux
```

4 Verify whether the PATH environment variable points to the base directory, where the SDK was installed.

To verify the path, run the **env | grep PATH** command and check whether the following path is displayed:

```
PATH=<sdk_extract>/sysroots/
x86_64-wrlinuxsdk-linux/usr/bin:
<sdk_extract>/sysroots/x86_64-wrlinuxsdk-linux/usr/bin/x86_64-wrs-linux
```

Alternatively, you can use the echo command:

```
echo $PATH
<sdk_extract>/sysroots/x86_64-wrlinuxsdk-linux/usr/bin:
<sdk_extract>/sysroots/x86_64-wrlinuxsdk-linux/usr/bin/x86_64-wrs-linux
```

5 Navigate to the directory that contains the application source code, and start building the application.

Note You should remove all the *.la temporary files from the SDK root file system. To do this, use the following commands:

```
bash# cd  <sdk_extract>/sysroots/
bash# find . -name \*.la | xargs rm -f
```

Build Native RPMs

This section describes the procedure for building applications by using either the native environment, or the cross-build environment. It is recommended that you use the native build environment.

There are two ways of building applications from source code.

One method is to build an application from a source code archive; this is explained in this section. The other method is to build it from a source RPM, which is not recommended.

Prerequisites

Ensure that the following requirements are met before you proceed:

- The application build environment has been set up to use either the native build environment, or the cross-build environment.
- You have read the README file to understand the build process for building the application.

Configuration Procedure

To build applications, use the following steps:

1 Navigate to the directory that contains the source code for the application.

2 Run the following commands to extract the application (if compressed).

```
bash-4.1$ tar xzvf tcpdump-4.7.4.tar.gz
```

3 Change your directory to the application directory.

```
bash-4.1$ cd tcpdump-4.7.4
```

4 Build your application to generate an executable file.

```
tcpdump-4.7.4$ ./configure
tcpdump-4.7.4$ make
```

5 Verify the executable file in your directory.

```
tcpdump-4.7.4$ ls -1 ./tcpdump
-rwxr-xr-x 1 john eng 3677288 Jun 15 23:10 ./tcpdump
```

The executable file is listed as `tcpdump`.

The executable file is ready to be packaged for hosting your application on IOS XR.

What to do Next

Package the application binaries so that it can be installed on IOS XR.

The recommended packaging format is RPM so that it can hosted on IOS XR.

To build an RPM:

- You need a .spec file.
- You must run the **rpmbuild** command.

Use the following steps to package the binaries:

1 Create a .spec file in the SPECS directory.

```
# %define __strip /bin/true

Name: tcpdump
Version: 4.7.4
Release: XR
Buildroot: %{_tmppath}/%{name}-%{version}-%{release}-root
License: Copyright (c) 2015 Cisco Systems Inc. All rights reserved.
Packager: mark
SOURCE0 : %{name}-%{version}.tar.gz
Group: 3'rd party applicaiton
Summary: Tcpdump cross compiled for WRL6

%description
This is a cross compiled version of tcpdump using IOS XR sdk for WRL7

%prep

%setup -q -n %{name}-%{version}

%build
  # This where sdk is being sourced
source /nobackup/mark/sdk_extract_18/tmp/env.sh
 ./configure
 make


 %install
rm -rf ${RPM_BUILD_ROOT}
 # make DESTDIR=$RPM_BUILD_ROOT install
mkdir -p ${RPM_BUILD_ROOT}%{_sbindir}
 install -m755 tcpdump ${RPM_BUILD_ROOT}%{_sbindir}
```

```
%files
 %defattr(-,root,root)
 %{_sbindir}/tcpdump

%pre

 %post

 %preun

 %postun

%clean
rm -rf $RPM_BUILD_ROOT
```

2 Build the RPM.

```
mark@tenby:redhat$ cd /usr/src/redhat/SPECS/
mark@tenby:SPECS$ rpmbuild -ba tcpdump.spec
```

The RPM build used is the 5.4.14 version.

3 Verify that the binary is built in the RPMS directory.

```
mark@tenby:x86_64$ pwd /usr/src/redhat/RPMS/x86_64

mark@tenby:x86_64$ ls

tcpdump-4.7.4-XR.x86_64.rpm
```

The native applications are ready to be hosted. For information on hosting native applications, see Run iPerf as a Native Application.