NX-SDK

This chapter contains the following topics:

About the NX-SDK

The Cisco NX-OS SDK (NX-SDK) is a C++ abstraction and plug-in library-layer that streamlines access to infrastructure for automation and custom native application creation, such as generating custom:

- CLIs
- Syslogs
- Event and Error managers
- Inter-application communication
- High availability (HA)
- Route manager

The NX-SDK also supports Python bindings.

For more information on Cisco NX-SDK, go to https://github.com/CiscoDevNet/NX-SDK were you can find release and documentation information. Click the `versions.md` link to get information on features and details on each supported release.

Requirements

The NX-SDK has the following requirements:

- Docker
- A Linux environment (either Ubuntu 14.04 or higher, or CentOS 6.7 or higher).
- Cisco SDK (optional) build environment.

Note The Cisco SDK is required to start applications in VSH. VSH requires that all applications be installed through RPMs, which requires that you build them in the Cisco SDK.

The Cisco SDK is not required for Python applications.

The Cisco SDK is not required for C++ application, but is still recommended. Using **g++** to build applications and then copying the built files to the switch may pose stability risks as **g++** is not supported.

Install the NX-SDK

Procedure

Step 1 **Note** The Cisco SDK is required for applications that are started in VSH.

The Cisco SDK is optional for applications that are started in Bash.

(Optional) Build the Cisco SDK RPM to persist on switch reloads and from standby mode.

a) Pull the Docker image for Ubuntu 14.04+ or Centos 6.7+ from https://hub.docker.com/r/dockercisco/nxsdk.

b) Source for a 32-bit environment:

Example:

```
export ENXOS_SDK_ROOT=/enxos-sdk
cd $ENXOS_SDK_Root
source environment-setup-x86-linux
```

Step 2 Clone the NX-SDK toolkit from https://github.com/CiscoDevNet/NX-SDK.git.

Example:

```
git clone  https://github.com/CiscoDevNet/NX-SDK.git
```

What to do next

You can find the following references to the API in `$PWD/nxsdk` and includes the following:

- The NX-SDK public C++ classes and APIs,
- Example applications, and
- Example Python applications.

Building and Packaging C++ Applications

The following instructions describe how to build and package your custom C++ NX-OS application.

Procedure

Step 1 Build your application files.

a) Building a C++ application requires adding your source files to the `Makefile`

Example:

The following example uses the `customCliApp.cpp` file from `/examples`.

```
...
##Directory Structure
...
EXNXSDK_BIN:= customCliApp
...
```

b) Build the C++ application using the**make** command.

Example:

```
$PWD/nxsdk# make clean
    $PWD/nxsdk# make all
```

Step 2 (Optional) Package your application.

Autogenerate RPM Package

Custom RPM packages for your applications must run on VSH and allow you to specify whether a given application persists on switch reloads or switchovers. Use the following to create a custom specification file for your application.

Note RPM packaging is required to be done within the provided ENXOS Docker image.

a) Use the rpm_gen.py script to auto-generate RPM package for a custom application.

Example:

Specify the -h option of the script to display the usages of the script.

```
/NX-SDK# python scripts/rpm_gen.py -h
```

b) By default, NXSDK_ROOT is set to /NX-SDK. If NX-SDK is installed in another location other than the default, then you must set NXSDK_ROOT env to the appropriate location for the script to run correctly.

Example:

```
export NXSDK_ROOT=<absolute-path-to-NX-SDK>
```

Example of autogenerate RPM package for C++ App examples/customCliApp.cpp

```
/NX-SDK/scripts# python rpm_gen.py CustomCliApp
##############################################################################################################

Generating rpm package...

Executing(%prep): /bin/sh -e /enxos-sdk/sysroots/x86_64-wrlinuxsdk-linux/var/tmp/rpm-tmp.49266
```

```
+ umask 022
+ cd /enxos-sdk/sysroots/x86_64-wrlinuxsdk-linux/usr/lib/rpm/../../src/rpm/BUILD
+ exit 0
Executing(%build): /bin/sh -e
/enxos-sdk/sysroots/x86_64-wrlinuxsdk-linux/var/tmp/rpm-tmp.49266
+ umask 022
+ cd /enxos-sdk/sysroots/x86_64-wrlinuxsdk-linux/usr/lib/rpm/../../src/rpm/BUILD
+ exit 0
Executing(%install): /bin/sh -e
/enxos-sdk/sysroots/x86_64-wrlinuxsdk-linux/var/tmp/rpm-tmp.49266
+ umask 022
+ cd /enxos-sdk/sysroots/x86_64-wrlinuxsdk-linux/usr/lib/rpm/../../src/rpm/BUILD
+ /bin/rm -rf
/enxos-sdk/sysroots/x86_64-wrlinuxsdk-linux/usr/lib/rpm/../../../var/tmp/customCliApp-root

+ /bin/mkdlr -p
/enxos-sdk/sysrOOts/x86_64-wrIinuxsdk-linux/usr/lib/rpm/../../../var/tmp/customCliApp-root//isan/bin

+ cp -R /NX-SDK/bin /enxos-sdk/sysroots/x86_64-wrlinuxsdk-linux/usr/lib/rpm/..
/../../var/tmp/customCliApp-root//isan/bin
+ exit 0
Processing files: customCliApp-1.0.x86_64
Requires: libc.so.6 libc.so.6(GLIBC 2.0) 3.0) Libc.so.6(GLIBC_2.1.3) libdl.so.2 libgcc_s.so.1
 libgcc_s.so.1(GCC_3.0) libm.so.6 libnxsdk.so libstdc++.so.6 libstdc++.so.6 (CXXAB1 1.3)
libstdc++.so.6(GLIBCXX 3.4) libstdc++.so.6(GLIBCXX_3.4.14) rt1d(GNU HASH)
Checking for unpackaged file(s):
/enxos-sdk/sysroots/x86_64-wrlinuxsdk-linux/usr/lib/rpm/check-files
/enos-sdk/sysroots/x86_64-wrlinuxsdk-linux/usr/lib/rpm/../../../var/tmp/customCliApp-root
Wrote: /enxos-sdk/sysrootS/X86_64-wrlinuxsdk-linux/usr/src/rpm/SRPMS/customCliApp-1.0.src-rpm

Wrote:
/enxos-sdk/sysrootS/X86_64-wrlinuxsdk-linux/usr/src/rpm/RPMS/x86_64/customCliApp-1.0.x86_64.rpm
Executing($clean): /bin/sh -e
/enxos-sdk/sysroots/x86_64-wrlinuxsdk-linux/var/tmp/rpm-tmp.49266
+ umask 022
+ cd /enxos-sdk/sysroots/x86_64-wrlinuxsdk-linux/usr/lib/rpm/../../src/rpm/BUILD
+ / bin/rm -rf
/enxos-sdk/sysroots/x86_64-wrlinuxsdk-linux/usr/lib/rpm/../../../var/tmp/customCliApp-root

RPM package has been built
##################################################################################################

SPEC file: /NX-SDK/rpm/SPECS/customCliApp.spec
RPM file : /NX-SDK/rpm/RPMS/customCliApp-1.0.x86_64.rpm
```

Manually generate RPM Package

Custom RPM packages for your applications are required to run on VSH and allow you to specify whether a given application persists on switch reloads or system switchovers. Use the following steps to create a custom specification file (`*.spec`) for your application.

a) Export the Cisco SDK RPM source to `$RPM_ROOT`.

Example:

```
export RPM_ROOT=$ENXOS_SDK_ROOT/sysroots/x86_64-wrlinuxsdk-linux/usr/src/rpm
```

b) Enter the `$RPM_ROOT` directory.

Example:

```
ls $RPM_ROOT (BUILD RPMS SOURCES SPECS SRPMS)
```

c) Create/edit your application-specific `*.spec` file.

Refer to the `customCliApp.spec` file in the `/rpm/SPECS` directory for an example specification file.

Note We recommend installing application files to `/isan/bin/nxsdk` on the switch as per the example `customCliApp.spec` file.

Example:

```
vi $RPM_ROOT/SPECS/<application>.spec
```

d) Build your RPM package.

Example:

```
rpm -ba $RPM_ROOT/SPECS/<application>.spec
```

A successful build generates an RPM file in `$RPMS_ROOT/RPMS/x86_64/`

Installing and Running Custom Applications

You can install applications by copying binaries to the switch, or installing unpacking the binaries from the RPM package.

Note Only custom applications that are installed from RPM packages can persist on switch reload or system switchovers. We recommend reserving copying binaries to the switch for simple testing purposes.

Before you begin

The switch must have the NX-SDK enabled before running any custom application. Run **feature nxsdk** on the switch.

Procedure

Step 1 Install your application using either VSH or Bash.

To install your application using VSH, perform the following:

a) Add the RPM package to the installer.

Example:

```
switch(config)# install add bootflash:<app-rpm-package>.rpm
```

b) After installation, check if the RPM is listed as inactive.

Example:

```
switch(config)# show install inactive
```

c) Activate the RPM package.

Example:

```
switch(config)# install activate <app-rpm-package>
```

d) After activation, check if the RPM is listed as active.

Example:

```
switch(config)# show install active
```

To install your application using Bash, run the following commands:

```
switch(config)# run bash sudo su
bash# yum install /bootflash/<app-rpm-package>.rpm
```

Step 2 Start your application.

C++ applications can run from VSH or Bash.

- To run a C++ application in VSH, run the **nxsdk** command:

  ```
  switch(config)# nxsdk service-name /<install directory>/<application>
  ```

  Note If the application is installed in /isan/bin/nxsdk, the full file path is not required. You can use the **nxsdk service-name** *app-name* form of the command.

- To run a C++ application in Bash, start Bash then start the application.

  ```
  switch(config)# run bash sudo su
  bash# <app-full-path> &
  ```

Python applications can run from VSH or Bash.

- To run a Python application from VSH, run the **nxsdk** command:

  ```
  switch(config)# nxsdk service-name <app-full-path>
  ```

  Note The Python application must be made executable to start from VSH:

  - Run the **chmod +x <app-full-path>** command.
  - Add `#!/isan/bin/nxpython` to the first link of your Python application.

- To run a Python application from Bash:

  ```
  switch(config)# run bash sudo su
  bash# /isan/bin/python <app-full-path>
  ```

  Note Use `/isan/bin/python` to run Python applications in Bash.

Step 3 Run **show nxsdk internal service** to verify that your application is running.

Example:

```
switch(config)# show nxsdk internal service

  NXSDK Started/Temp unavailabe/Max services : 2/0/32
  NXSDK Default App Path         : /isan/bin/nxsdk
  NXSDK Supported Versions       :  1.0 1.5 1.7.5

  Service-name              Base App        Started(PID)    Version    RPM Package
  ------------------------- --------------- ------------    ---------- --------------------

  /isan/bin/capp1           nxsdk_app2      VSH(25270)      1.7.5        capp1-1.0.x86_64
  /isan/bin/TestApp.py      nxsdk_app3      BASH(27823)     -          -
```

Step 4 Stop your application.

You can stop your application in the following ways:

- To stop all NX-SDK applications, run the **no feature nxsdk** command.
- To stop a specific application in VSH, run:

  switch(config)# no nxsdk service-name /<install directory>/<application>
- To stop a specific application in Bash, run:

  bash# <application> stop-event-loop

Step 5 Uninstall your application.

To uninstall the RPM from the switch using VSH, perform the following:

a) Deactivate the active RPM package.

Example:

```
switch# install deactive <app-rpm-package>
```

b) Verify that the package is deactivated.

Example:

```
switch# show install inactive
```

c) Remove the RPM package.

Example:

```
switch# install remove <app-rpm-package>
```

To uninstall the RPM from the switch using Bash, run the `yum remove <app-full-path>` command.