NX-SDK

NX-SDK について

Cisco NX-OS SDK（NX-SDK）は、自動化およびカスタム アプリケーションの作成（カスタムの生成など）のためのインフラストラクチャへのアクセスを合理化する C++ 抽象化およびプラグイン ライブラリ レイヤです。

- CLI
- Syslog
- イベント マネージャとエラー マネージャ
- アプリケーション間通信
- ハイ アベイラビリティ（HA）
- ルート マネージャ

NX-SDK を使用したアプリケーション開発には、C++、Python、または Go を使用できます。

要件

NX-SDK では次の要件があります。

- Docker
- Linux 環境（Ubuntu 14.04 または Centos 6.7 のいずれか）。提供されている NX-SDK Docker コンテナを使用することをお勧めします。詳細については、「Cisco DevNet NX-SDK」を参照してください。

関連情報

Cisco NX-SDK の詳細については、次にアクセスしてください。

- Cisco DevNet NX-SDK。サポートされるリリースごとの機能と詳細については、`versions.md` リンク（https://github.com/CiscoDevNet/NX-SDK/blob/master/versions.md）を参照してください。

NX-SDK のインストール

手順

ステップ 1 （注） Cisco SDK は、VSH で起動されるアプリケーションに必要です。

Cisco SDK は、Bash で起動されたアプリケーションではオプションです。

（任意） スイッチのリロード時およびスタンバイモードから永続化する Cisco SDK RPM を構築します。

a) https://hub.docker.com/r/dockercisco/nxsdkから Ubuntu 14.04 以降または Centos 6.7 以降の Docker イメージをプルします。

b) 32 ビット環境の送信元：

例：

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

ステップ 2 https://github.com/CiscoDevNet/NX-SDK.git から NX-SDK ツールキットのクローンを作成します。

例：

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

次のタスク

API への次の参照は、`$PWD/nxsdk` にあり、次のものが含まれます：

- NX-SDK パブリック C++ クラスおよび API、
- アプリケーション例と
- Python アプリケーションの例。

C++ アプリケーションの構築とパッケージ化

次の手順では、カスタム C++ NX-OS アプリケーションを構築およびパッケージ化する方法について説明します。

手順

ステップ 1 アプリケーション ファイルの構築

a) C++アプリケーションを構築するには、送信元ファイルを`Makefile`に追加する必要があります。

例：

次の例では、`/examples` の `customCliApp.cpp` ファイルを使用します。

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

b) **make** コマンドを使用して C++ アプリケーションを構築します。

例：

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

ステップ 2 （任意） アプリケーションをパッケージします。

[RPM パッケージの自動生成（Auto-generate RPM package）]

アプリケーションのカスタム RPM パッケージは VSH で実行する必要があり、特定のアプリケーションがスイッチのリロードまたはシステム スイッチオーバーで保持されるかどうかを指定できます。以下を使用して、アプリケーションのカスタム仕様ファイルを作成します。

（注） RPM パッケージ化は、提供されている ENXOS Docker イメージ内で実行する必要があります。

a) rpm_gen.py スクリプトを使用して、カスタムアプリケーションの RPM パッケージを自動生成します。

例：

スクリプトの使用状況を表示するには、スクリプトの -h オプションを指定します。

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

b) デフォルトでは、NXSDK_ROOT は /NX-SDK に設定されます。NX-SDK がデフォルト以外の別の場所にインストールされている場合は、スクリプトが正しく実行されるように NXSDK_ROOT 環境を適切な場所に設定する必要があります。

例：

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

C++ アプリケーション用の RPM パッケージの自動生成の例 example/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-1.0-7.03.I6.1.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) rtld(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-1.0-7.0.3.I6.1.src-rpm

Wrote:
/enxos-sdk/sysrootS/X86_64-wrlinuxsdk-linux/usr/src/rpm/RPMS/x86_64/customCliApp-1.0-7.0.3.I6.1.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-1.0-7.0.3.I6.1.x86_64.rpm
```

[RPM パッケージの手動生成（Manually-generate RPM Package）]

アプリケーションのカスタム RPM パッケージは VSH で実行する必要があり、特定のアプリケーションがスイッチのリロードまたはシステムスイッチオーバーで保持されるかどうかを指定できます。次の手順を使用して、アプリケーションのカスタム仕様ファイル（`*.spec`）を作成します。

a) Cisco SDK RPM 送信元を `$RPM_ROOT` にエクスポートします。

例：

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

b) `$RPM_ROOT` ディレクトリに移動します。

例：

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

c) アプリケーション固有の `*.spec` ファイルを作成/編集します。

仕様ファイルの例については、`/rpm/SPECS` ディレクトリの `customCliApp.spec` ファイルを参照してください。

（注） `customCliApp.spec` ファイルの例に従って、アプリケーション ファイルをスイッチの `/isan/bin/nxsdk` にインストールすることを推奨します。

例：

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

d) RPM パッケージを構築します。

例：

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

構築を成功させることで`$RPMS_ROOT/RPMS/x86_64/` に RPM ファイルが生成されます。

カスタム アプリケーションのインストールと実行

バイナリをスイッチにコピーするか、または RPM パッケージからバイナリを解凍してインストールすることで、アプリケーションをインストールできます。

（注） RPM パッケージからインストールされたカスタムアプリケーションだけが、スイッチのリロードまたはシステムのスイッチオーバー時に保持できます。単純なテストの目的で、スイッチへのバイナリのコピーを予約することを推奨します。

スイッチ（オンボックス）内で NX-SDK アプリケーションを実行するには、Cisco SDK ビルド環境がインストールされている必要があります。

（注） VSH でアプリケーションを起動するには、Cisco SDK が必要です：VSH では、すべてのアプリケーションを RPM を介してインストールする必要があります。そのためには、Cisco SDK に組み込まれている必要があります。

Python アプリケーションには Cisco SDK は必要ありません。

Cisco SDK は C++ アプリケーションには必要ありませんが、推奨されます：**g++** を使用してアプリケーションを構築し、構築されたファイルをスイッチにコピーすると、**g++** がサポートされていないため、安定性が低下する可能性があります。

スイッチにカスタムアプリケーションをインストールまたは実行するには、次の手順を実行します。

始める前に

カスタム アプリケーションを実行する前に、スイッチで NX-SDK を有効にする必要があります。スイッチ上で**feature nxsdk**を実行します。

手順

ステップ 1 VSH または Bash を使用してアプリケーションをインストールします。

VSH を使用してアプリケーションをインストールするには、次の手順を実行します。

a) インストーラに RPM パッケージを追加します。

例：

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

b) インストール後、RPM が非アクティブとしてリストされているかどうかを確認します。

例：

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

c) RPM パッケージを現用系にします。

例：

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

d) アクティブ化後、RPM が現用系としてリストされているかどうかを確認します。

例：

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

Bash を使用してアプリケーションをインストールするには、次のコマンドを実行します。

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

ステップ 2 アプリケーションを起動します。

C++ アプリケーションは、VSH または Bash から実行できます。

- VSH で C++ アプリケーションを実行するには、次の **nxsdk** コマンドを実行します。

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

（注） アプリケーションが /isan/bin/nxsdk にインストールされている場合、完全なファイルパスは必要ありません。コマンド **nxsdk service-name** *app-name* 形式を使用できます。

- Bash で C++ アプリケーションを実行するには、Bash を起動してからアプリケーションを起動します。

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

Python アプリケーションは、VSH または Bash から実行できます。

- VSH から Python アプリケーションを実行するには、次の**nxsdk**コマンドを実行します：

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

（注） VSH から起動するには、Python アプリケーションを実行可能にする必要があります：

  - **chmod +x** *app-full-path*を実行します。
  - `#!/isan/bin/nxpython` を Python アプリケーションの最初のリンクに追加します。

- Bash から Python アプリケーションを実行するには、

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

（注） デフォルトでは、`NX-SDK` は `/isan/bin/nxsdk` を使用して Bash で Python アプリケーションを実行しますが、必要に応じて別のインストールディレクトリを指定できます。

ステップ 3 **show nxsdk internal service** を実行して、アプリケーションが実行していることを確認します

例：

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

switch(config)# show nxsdk internal service

  NXSDK total services (Max Allowed) : 2 (32)
  NXSDK Default App Path         : /isan/bin/nxsdk
  NXSDK Supported Versions       :  1.0

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

  /isan/bin/capp1           nxsdk_app2      VSH(25270)      1.0
capp1-1.0-7.0.3.I6.1.x86_64
  /isan/bin/TestApp.py      nxsdk_app3      BASH(27823)     -          -
```

ステップ 4 アプリケーションを停止します。

次の方法でアプリケーションを停止できます。

- すべての NX-SDK アプリケーションを停止するには、**no feature nxsdk**を実行します。
- VSH で特定のアプリケーションを停止するには、 **no nxsdk service-name** */install directory/application*を実行します。
- Bash で特定のアプリケーションを停止するには、*application* **stop-event-loop**を実行します。

ステップ 5 アプリケーションをアンインストールします。

VSH を使用してスイッチから RPM をアンインストールするには、次の手順を実行します。

a) アクティブな RPM パッケージを非アクティブ化します。

例：

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

b) パッケージが非アクティブ化されていることを確認します。

例：

```
switch# show install inactive
```

c) RPM パッケージを削除します。

例：

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

Bash を使用してスイッチから RPM をアンインストールするには、**yum remove** *app-full-path*を実行します。

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。