NX-SDK

This chapter contains the following sections:

About the NX-SDK

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

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

You can use C++, Python, or Go (with some exceptions) for application development with NX-SDK.

Requirements

The NX-SDK has the following requirements:

- Docker
- A Linux environment (either Ubuntu 14.04, or Centos 6.7). Cisco recommends using the provided NX-SDK Docker containers. For more information, see Cisco DevNet NX-SDK.

Support for Local (On Switch) and Remote (Off Switch) Applications

Applications that are developed with NX-SDK are created or developed off of the switch in the Docker containers that the NX-SDK provides. After the application is created, you have flexibility of where the applications can be deployed:

- Local (on-box) applications run on the switch. For information, see About On-Box (Local) Applications, on page 2
- Remote (off-box) applications run off switch. This option, supported with NX-SDK 2.0 and later, enables you to deploy the application to run anywhere other than on the switch. For information, see About Remote Applications, on page 4.

Related Information

For more information about Cisco NX-SDK, go to:

- Cisco DevNet NX-SDK. Click the `versions.md` link (Cisco DevNet NX-SDK Versions) to get information about features and details on each supported release.
- NX-SDK Readmes

As needed, Cisco will add information for NX-SDK to github.

Considerations for Go Bindings

Go bindings are supported at various levels depending on the release of NX-SDK and whether apps are running locally or remotely.

- Go bindings for any version of NX-SDK remote application are pre-EFT quality.
- Go bindings for a local NX-SDK 2.0 application is pre-EFT.
- Go bindings for a local NX-SDK 1.7.5 application or earlier is supported.

For more information, see GO Bindings for NX-SDK Applications.

About On-Box (Local) Applications

With on box (local) applications, you install the NX-SDK, build your application in whichever supported language you choose, package the app as an `.rpm` file which can be installed on the switch, then install and run your applications on the switch. The `.rpm` files can be manually generated or autogenerated.

Application development occurs in the containers that are provided by NX-SDK. You will use a different container and tools for local applications than remote applications. For more information, see Default Docker Images, on page 2.

For information about building, installing, and running local applications, see Cisco DevNet NX-SDK .

Default Docker Images

NX-SDK has the following Docker images and tools by default for local or remote use.

Usage	Contents
On Switch	Cisco ENXOS SDK Wind River Linux (WRL) tool chain for cross compiling Multi-language binding toolkit Beginning with NX-SDK 1.75, a Go compiler
Off switch (remote)	NX-SDK multi-language binding Toolkit with pre-built libnxsdk.so A Go compiler RapidJSON gRPC for remote API support

For more information, see https://github.com/CiscoDevNet/NX-SDK#readmehttps://github.com/CiscoDevNet/NX-SDK/tree/master/readmes/nxsdk_docker_images.md.

Guidelines and Limitations for NX-SDK

NX-SDK has usage guidelines and limitations for running applications locally (on box) or remotely (off box).

For guidelines and limitations, see "Helpful Notes" at Cisco DevNet NX-SDK .

- For remote applications, ports that connect to the SDK server are from 50002 through 50100. Make sure that these ports are open and not used by other services. Port 50051 is blocked for use by an internal application, and cannot be used by remote applications.
- The following limitations apply to developing a custom application through NX-SDK:
  - You must develop applications with any current Linux distribution that have GNU C/C++ toolchains and standard libraries.
  - We recommend developing applications in the provided Docker images. Run the Docker image and associated tools requires a minimum of 8 MB of free memory and 64-bit hosts. See Default Docker Images, on page 2.
- Cisco recommends that you build and test your applications in Bash. When you deploy them in production, build the applications as RPMs and run them in the NX-OS VSH (Vshell).
- Cisco Nexus switches support a maximum of 32 applications in Bash and VSH.
- For NX-SDK 2.0, there is a 1-to-1 limit for remote NX-SDK applications and NX-SDK servers running in the Cisco Nexus switch.
- A maximum of 10 remote NX-SDK applications can run simultaneously in a switch.
- The NX-SDK server accepts remote requests only from a CLI-configured application and not from random applications.
- For any error that occurs, the SDK server throws an exception.
- Some APIs do not run in a remote application. Refer to API documentation for more information.

- Remote client library code is not open source, so NX-SDK remote apps need to run in the NX-SDK remote docker container. To use remote NX-SDK in your own OS, make a request in github with your OS and compiler version. Cisco can generate a remote libnxsdk.so based on your version.

Off-Box (Remote) Applications

About NX-SDK 2.0

The NX-SDK version 2.0 enables execution-environment flexibility for developers to run their applications wherever needed. With this version of NX-SDK, your applications are still developed off the switch in containers, but you can run the apps either on the switch or off the switch, for example in a cloud.

NX-SDK 2.0 offers the following benefits:

- Easy integration of the switch into the customer environment.
- Scalability to enable the switch to seamlessly operate in data centers, public clouds, and private clouds.
- Decoupling customer apps from switch resources so that changes at the switch-level resources do not require change or rewrite of applications.
- Single library with simple to use APIs for applications to link against, which simplifies switch interactions and allows applications to be written in high-level languages that are easier to write and debug.
- Running Remote services are more secure than on-box applications.

For more information, see https://github.com/CiscoDevNet/NX-SDK/blob/master/readmes/NXSDK_in_NXOS.mdDeploying NX-SDK Applications Remotely.

About Remote Applications

Remote applications can be on a different switch that is not a Cisco Nexus switch. Remote, or off-box, applications call through the NX-SDK layer to interact with the switch to read information (get) or write information (set).

Both local and remote NX-SDK applications use the same APIs, which offer you the flexibility to deploy NX-SDK applications on- or off-box.

To run remotely, an application must meet specific requirements. For information, see https://github.com/CiscoDevNet/NX-SDK/blob/master/readmes/NXSDK_in_NXOS.mdDeploying NX-SDK Applications Remotely.

Backward Compatibility for Pre-2.0 NX-SDK Applications

NX-SDK 2.0 has conditional backward compatibility for NX-SDK v1.75 applications depending on how these applications were developed:

- Usually, NX-SDK supports remotely running an app that you created before NX-SDK 2.0 without requiring you to completely rewrite your app. Instead, you can reuse the same app without modifying it to change the API calls. To support older apps in the new NX-SDK 2.0 model, the API call must provide IP and Port parameters. These parameters are not available in NX-SDK 1.75 and earlier, but you can add the IP address and Port information as environment variables that the app can export to the SDK server.

- However, sometimes backward compatibility for pre-NX-SDK 2.0 apps might not be supported. It is possible that some APIs in older apps might not support, or be capable of, running remotely. In this case, the APIs can throw an exception. Depending on how complete and robust the exception-handling is for the original application, the application might operate unpredictably, and in worst cases, possibly crash.

For more information, see https://github.com/CiscoDevNet/NX-SDK/blob/master/readmes/NXSDK_in_NXOS.mdDeploying NX-SDK Applications Remotely.

NX-SDK Security

Beginning with NX-OS 9.3(1), NX-SDK 2.0 supports the following security features:

- Session security. Remote applications can connect to the NX SDK server on the switch through Transport Layer Service (TLS) to provide encrypted sessions between the applications and the switch's NX SDK server.
- Server certificate security. For new switch deployments with Cisco NX-OS 9.3(1), the NX-SDK server generates a one-day temporary certificate to provide enough time to install a custom certificate.

  If your NX-SDK server already has a custom certificate that is installed, for example, if you are upgrading from a previous NX-SDK version to NX-SDK 2.0, your existing certificate is retained and used after upgrade.
- API write-call control. NX-SDK 2.0 introduces security profiles, which enable you to select a pre-defined policy for controlling how much control an application has with the NX-SDK server. For more information about security profiles, see Security Profiles for NX SDK 2.0, on page 5.

Security Profiles for NX SDK 2.0

In previous releases, the APIs for SDK version 1.75 were permitted only to read and get data for events. Beginning in Cisco NX-OS Release 9.3(1), NX-SDK 2.0 supports different types of operations, including write calls.

The ability of an app to read or write to the switch can be controlled through a security profile. A security profile is an optional object that is attached to the applications' service running in the switch. Security profiles control an application's ability to write to the switch, and in turn, control the applications ability to modify, delete, or configure switch functionality. By default, application writes are disallowed, so for each application, you will need to create a security profile that enables write access to the switch.

Cisco's NX-SDK offers the following security profiles.

Profile	Description	Values
Deny	Prevents any API calls from writing to the switch except for adding CLIs.	This is the default profile.

Profile	Description	Values
Throttle	Allows APIs that modify the switch, but only up to a specified number of calls. This security profile applies throttling to control the number of API calls. The application is allowed to write up to the limit, but when the limit is exceeded, writing stops, and the reply sends an error message.	The throttle is 50 API calls, and the throttle resets after five seconds.
Permit	APIs that modify the switch are allowed without restriction	

For more information about security profiles in NX-SDK, see Security Profiles for NX-SDK Applications .

For additional information about building, installing, and running applications, go to CiscoDevNet NX-SDK .