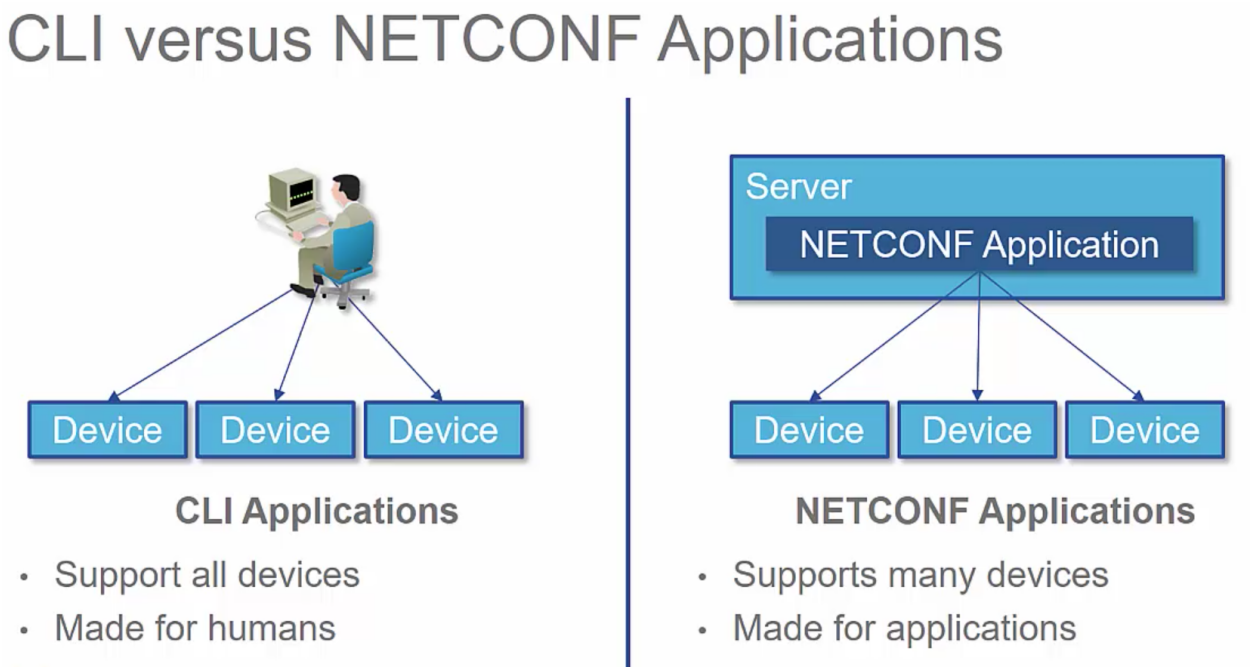
CLI vs. NETCONF



CLI scripting was the primary approach to making automated configuration changes to the network prior to NETCONF. CLI scripting has several limitations including lack of transaction management, no structured error management, and ever changing structure and syntax of commands that makes scripts fragile and costly to maintain.

NETCONF is a protocol that is defined by the IETF to address some of the challenges with CLI scripting by providing a standards-based method to "install, manipulate, and delete the configuration of network devices." NETCONF typically uses SSH for secure transport and XML to exchange data. NETCONF provides remote procedure calls (RPCs) to operate network devices in a programmatic fashion. In order for NETCONF to be useful as a network-wide protocol, it must have a common data model. YANG is a modeling language defined in IETF RFC 6020. YANG is used by NETCONF to define the objects and data in requests and replies.

RESTCONF is a protocol that exposes a REST-like interface to network devices and uses HTTP for transport and JSON or XML to interface with network devices exposing data modeled in YANG. Both RESTCONF and NETCONF enable access to the same YANG modeled data, but they use different API bindings. Note that RESTCONF does not replace NETCONF. RESTCONF just provides another mechanism to expose the same data modeled via YANG (such as configuration or operational data). Programmers or designers may choose which API to leverage based on experience, skill set, and requirements.

Your application will be built differently based on your desired mechanism for communicating with the device. Several protocols are available, including CLI, NETCONF, RESTCONF, and SNMP.

CLI-based communication has the following characteristics and considerations:

CLI interfaces exist on all devices, specifically to allow manual configuration by humans. What this means is that every device will have the capability to be programmed by applications such as yours.

Since CLI interfaces are the tool for performing configuration operations on devices, they provide access to just about every function supported by the device. Hence, via CLI, you are guaranteed to be able to do just about anything you desire.

The issue that is related to programmability via CLIs is that the data, both for setting configuration on the device, and for reading configuration or state, is unstructured. Since it is meant to be read and input by humans, there are no formatting rules other than what is determined by the creator of the CLI interfaces themselves. Humans are very adaptable and can make sense out of information in just about any format, but not so with a computer application. Hence, making this work in a programmatic sense can be challenging.

Network programmability developers are subject to differences in CLIs between products from different vendors. It is not just different vendor equipment either; quite often, different product families within a single vendor can have differences. Even different versions within a product family may have changed the CLI, usually for human readability or to accommodate a new feature. All these differences are the bane of the network programmer attempting to use CLIs to configure devices.

NX-API: One important type of CLI-based application, which improves on traditional CLI programming, is what has been created for Nexus devices. For these devices, there is an API that has been created called "NX-API," which is a REST-based CLI interface for these devices. The advantage here is that the developer avoids many of the issues that are related to CLI programming – the unstructured input and output become structured, and are far less susceptible to minor changes and variations in commands or output.

Documentation about the RESTful NX-API interface can be found online at developer.cisco.com.

NETCONF-based communication, on the other hand, has a different set of characteristics and considerations.

The most immediate issue with NETCONF (and similarly with SNMP or TL1) is that not all devices support NETCONF. So only a subset of the devices in your network may support the protocol. It is wise to consider whether NETCONF is supported in the devices you wish to control, before beginning development of your application.

Even if NETCONF is supported, this fact does not insure that your devices support the features you are looking for. So, it is wise to fully understand the complete set of YANG models that are supported by the devices you wish to control with your application.

The good news is that NETCONF is not subject to the differences, changes, and vagaries of a CLI. Therefore, if NETCONF is supported, and your YANG data models are likewise supported, you should be able to create applications which are consistent and which work consistently over time.

The main issue that is related to NETCONF/YANG environments is the actual defined data model, created via YANG. Different devices support different YANG models for the same or similar operations. If the YANG models are different for the devices in your domain of control, your application will have to make appropriate allowances.