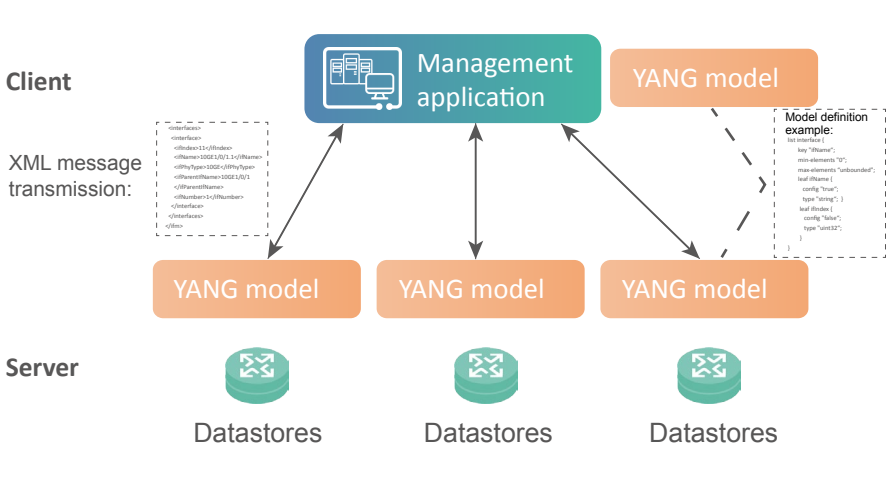
HUAWEI

NETCONF&YANG

What Are NETCONF and YANG

Network Configuration Protocol (NETCONF) is a network device management protocol that is similar to SNMP. NETCONF provides a framework for users to add, modify, or delete network device configurations, or query configurations, status, and statistics.



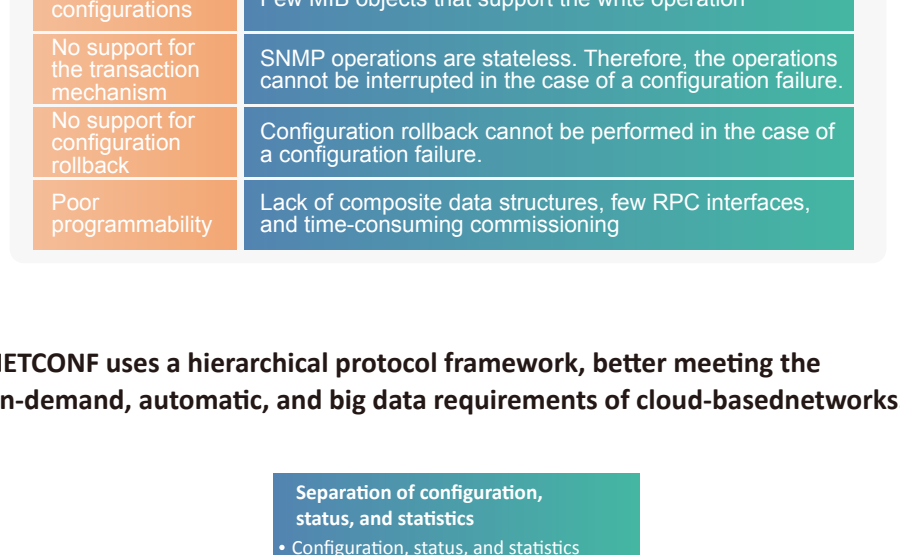
Similar to SNMP that uses MIB files to model data, NETCONF uses Yet Another Next Generation (YANG) as a data modeling language to describe the interaction models between the NETCONF client and server.

Comparison Item	SNMP	NETCONF	CLI
Defined by	IETF	IETF	None
Resource form	OID	XPATH	None
Data model file	.mib	.yang	No model
Modeling language	SMI	YANG	None
Operation management protocol	SNMP	NETCONF	None
Coding mode	BER	XML	Pure text
Transmission protocol	UDP	SSH	TELNET/SSH

Why Are NETCONF and YANG Required

One of the key network requirements of the cloud era is network automation, which includes quick, automatic, and on-demand service provisioning and automatic O&M. The traditional CLI mode and SNMP do not meet the requirements of cloud-based networks due to the following disadvantage:

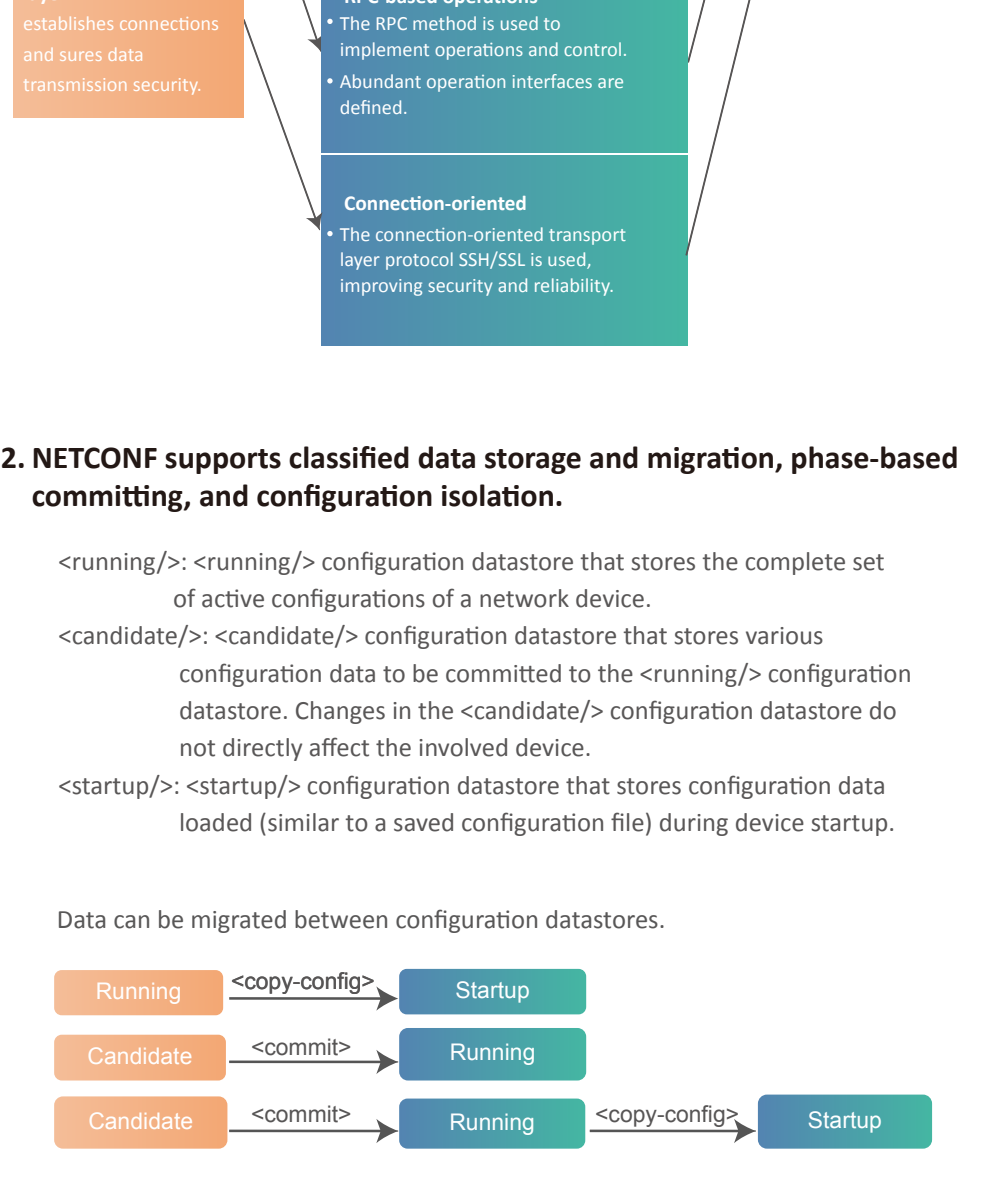
The traditional CLI mode is based on man-machine interfaces. The configuration is complex and varies with vendors. Therefore, the manual learning cost is high.



The SNMP configuration efficiency is low, and the transaction mechanism is not supported. Therefore, SNMP is often used for monitoring.

Poor reliability	Digital index (1.3.6.1.2.1.2.2.1.4)
Insufficient performance	Data configuration and reading are low, especially in the deployment of large-scale networks.
Difficult to deliver configurations	Few MIB objects that support the write operation
No support for the transaction mechanism	SNMP operations are stateless. Therefore, the operations cannot be interrupted in the case of a configuration failure.
No support for configuration rollback	Configuration rollback cannot be performed in the case of a configuration failure.
Poor programmability	Lack of composite data structures, few RPC interfaces, and time-consuming commissioning

1. NETCONF uses a hierarchical protocol framework, better meeting the on-demand, automatic, and big data requirements of cloud-basednetworks.



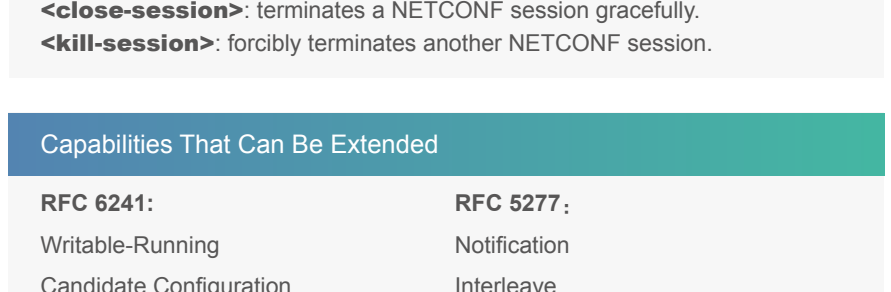
2. NETCONF supports classified data storage and migration, phase-based committing, and configuration isolation.

<running/>: <running/> configuration datastore that stores the complete set of active configurations of a network device.

<candidate/>: <candidate/> configuration datastore that stores various configuration data to be committed to the <running/> configuration datastore. Changes in the <candidate/> configuration datastore do not directly affect the involved device.

<startup/>: <startup/> configuration datastore that stores configuration data loaded (similar to a saved configuration file) during device startup.

Data can be migrated between configuration datastores.



3. NETCONF defines abundant operation interfaces and supports extension based on capability sets.

Basic Operations Supported by NETCONF (RFC 6241)

<get>: obtains part or all of the running configuration data and status data from the <running/> configuration datastore.
<get-config>: obtains configuration data.
<edit-config>: creates, modifies, or deletes configuration data.
<copy-config>: replaces a configuration datastore with the contents of another complete configuration datastore.
<delete-config>: deletes all data in a non-running configuration datastore.
<lock>: locks the configuration datastore of a device. A locked configuration datastore cannot be modified by other NETCONF users.
<unlock>: unlocks the configuration datastore of a device.
<close-session>: terminates a NETCONF session gracefully.
<kill-session>: forcibly terminates another NETCONF session.

Capabilities That Can Be Extended

RFC 6241:
Writable-Running
Candidate Configuration
Confirmed Commit
Rollback-on-Error
Validate
Startup
URL
XPath

RFC 5277:
Notification
Interleave

RFC 6243:
with-defaults

RFC 6022:
Iletf-netconf-monitoring

4. NETCONF operations are performed based on the YANG model.

The YANG model defines device function configuration templates. Compared with the CLI mode, the YANG model has the following advantages:

- Rich definition: Various basic data types and data attributes can be defined.
- Machine language: Structured definition is adopted, allowing users to define constraints that can be directly identified by the computer without manual intervention.
- Good scalability: The grouping, refine, augment, and typedef statements are supported for users to extend YANG models and data types.
- Easy integration: The IETF defines multiple YANG models and data types for reference.

Command line example:

ip address ip-address { mask | mask-length }

→ Information to be manually entered:

ip address 1.1.1.1 24

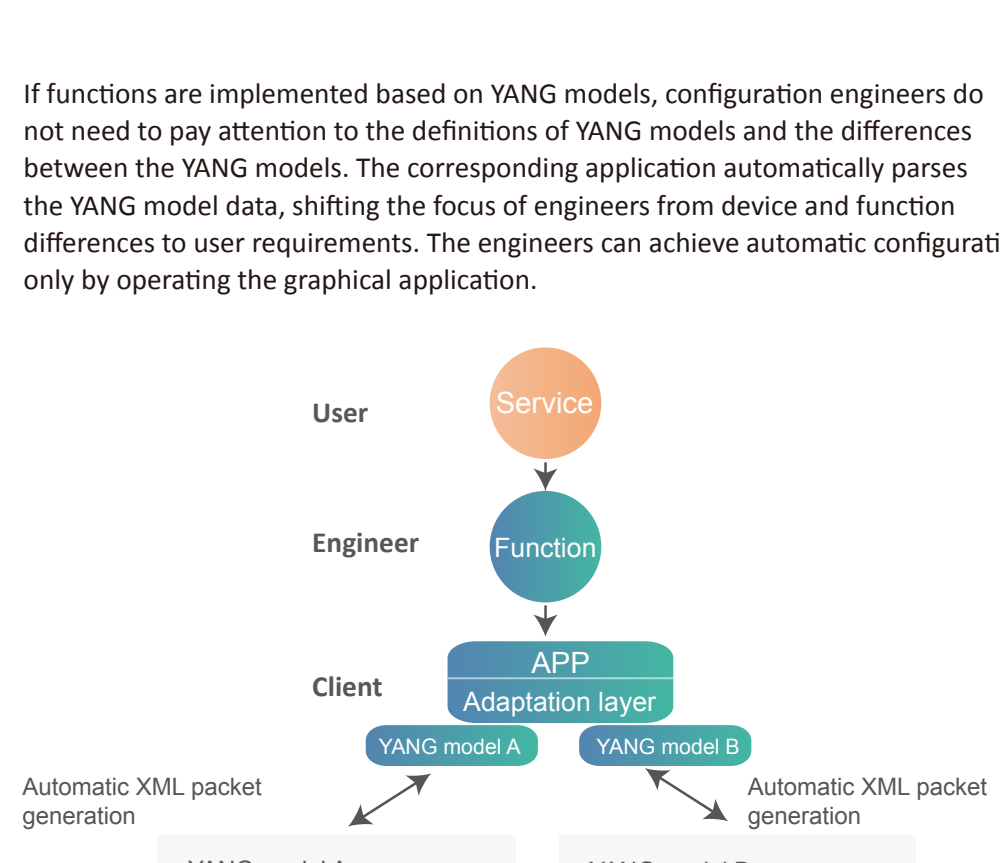
YANG model example:

```
{ list ip
  when "not(../../addrCfgType='negotiation')";
  key ip-address
    { leaf ip-address
          type inet:ipv4-address-no-zone
      leaf mask
          type ifmIpv4Mask}
}
```

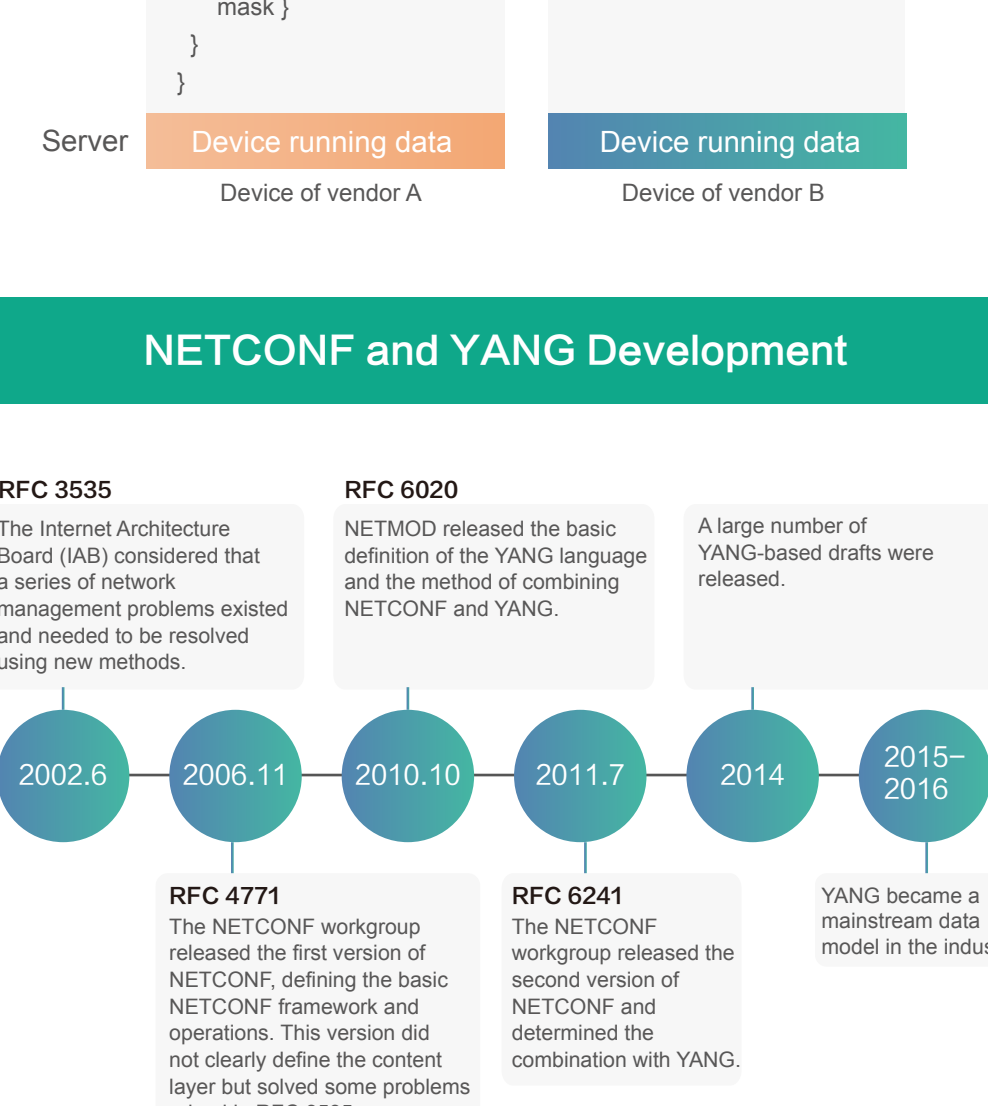
→ XML request packet sent by the client:

```
<ip xmlns="http://www.huawei.com/
   netconf/vrp/huawei-ip">
  <ip-address>1.1.1.1</ip-address>
   <mask>24</mask>
 </ip>
```

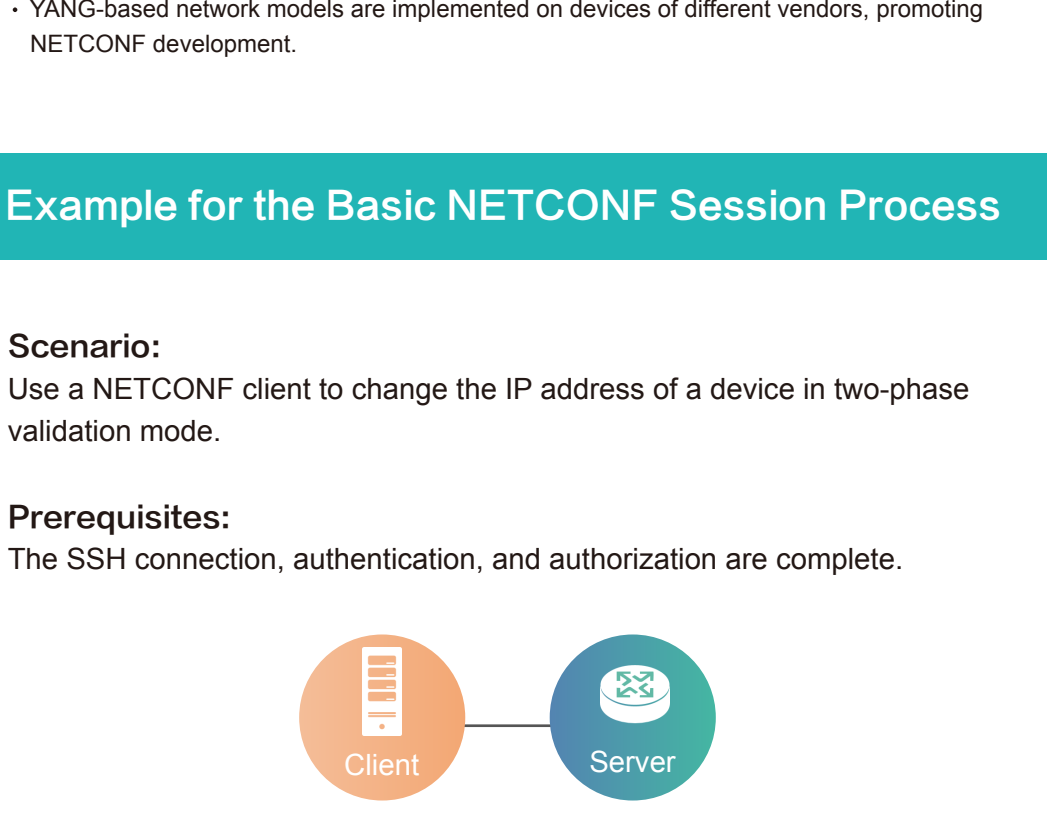
If functions are implemented in different ways on devices from different vendors: In traditional CLI mode, engineers need to learn the differences between command lines of different vendors and manually map the command lines.



If functions are implemented based on YANG models, configuration engineers do not need to pay attention to the definitions of YANG models and the differences between the YANG models. The corresponding application automatically parses the YANG model data, shifting the focus of engineers from device and function differences to user requirements. The engineers can achieve automatic configuration only by operating the graphical application.



NETCONF and YANG Development



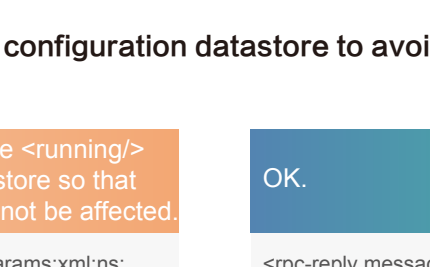
YANG drives the development of NETCONF.

- Universal models are defined based on YANG, making a breakthrough in NETCONF implementation and laying the foundation for model unification between vendors.
- YANG-based network models are implemented on devices of different vendors, promoting NETCONF development.

Example for the Basic NETCONF Session Process

Scenario:
Use a NETCONF client to change the IP address of a device in two-phase validation mode.

Prerequisites:
The SSH connection, authentication, and authorization are complete.



Step 1:
Establish a NETCONF session and advertise capabilities through<hello> messages.

Hello. I support the following capabilities:

```
<hello xmlns="urn:ietf:params:xml:ns:
netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:
base:1.1</capability>
  </capabilities>
</hello>
```

Hello. I support the following capabilities:

```
<hello xmlns="urn:ietf:params:xml:ns:
netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:
base:1.1</capability>
  </capabilities>
<session-id>1</session-id>
</hello>
```

Step 2:
Lock the <running/> configuration datastore to avoid conflicts with other clients.

I request to lock the <running/> configuration datastore so that my operations will not be affected.

```
<rpc xmlns="urn:ietf:params:xml:ns:
netconf:base:1.0" message-id="2">
  <lock>
    <target>
      <running></running>
    </target>
  </lock>
</rpc>
```

OK.

```
<rpc-reply message-id="2" xmlns="
urn:ietf:params:xml:ns:netconf:
base:1.0">
  <ok/>
</rpc-reply>
```

Step 3:
Copy the data in the <running/> configuration datastore to the <candidate/> configuration datastore to ensure that the configurations are the latest.

I request to copy data.

```
<rpc xmlns="urn:ietf:params:xml:ns:
netconf:base:1.0" message-id="3">
  <copy-config>
<target>
  <candidate></candidate>
</target>
<source>
  <running></running>
</source>
  </copy-config>
</rpc>
```

OK.

```
<rpc-reply message-id="3" xmlns=
"urn:ietf:params:xml:ns:netconf:
base:1.0">
  <ok/>
</rpc-reply>
```

Step 4: Edit configurations in the <candidate/> configuration datastore.

I request to configure 1.1.1.1 as the IP address of GE 1/0/0.

```
  <interfaces>
    <interface>
      <ifIndex>11</ifIndex>
      <ifName>10GE1/0/1.1</ifName>
      <ifPhyType>10GE</ifPhyType>
      <ifParentIfName>10GE1/0/1
      </ifParentIfName>
      <ifNumber>1</ifNumber>
    </interface>
  </interfaces>
</ifm>
```

OK.

```
list interface {
      key "ifName";
      min-elements "0";
      max-elements "unbounded";
      leaf ifName {
        config "true";
        type "string";  }
      leaf ifIndex {
        config "false";
        type "uint32";
      }
}
```

Step 5:
Commit configurations in the <candidate/> configuration datastore to the <running/> configuration datastore.

I request to commit configurations.

```
<rpc xmlns="urn:ietf:params:xml:ns:
netconf:base:1.0" message-id="6">
  <commit></commit>
</rpc>
```

OK. The configurations have been committed to the <running/> configuration datastore.

```
<rpc-reply message-id="6" xmlns=
"urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Step 6:
Unlock the <running/> configuration datastore.

The configuration is complete, and the <running/> configuration datastore can be unlocked.

```
<rpc xmlns="urn:ietf:params:xml:ns:
netconf:base:1.0" message-id="7">
  <unlock>
    <target>
      <running></running>
    </target>
  </unlock>
</rpc>
```

OK.

```
<rpc-reply message-id="7" xmlns="urn:ietf:
params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Follow-up procedure: Terminate the NETCONF session and tear down the SSH connection.

Acronyms and Abbreviations

NETCONF	Network Configuration Protocol
XML	Extensible Markup Language
RPC	Remote Procedure Call
YANG	Yet Another Next Generation