NETCONF by Example

v0.1.1 (2015-11-05)

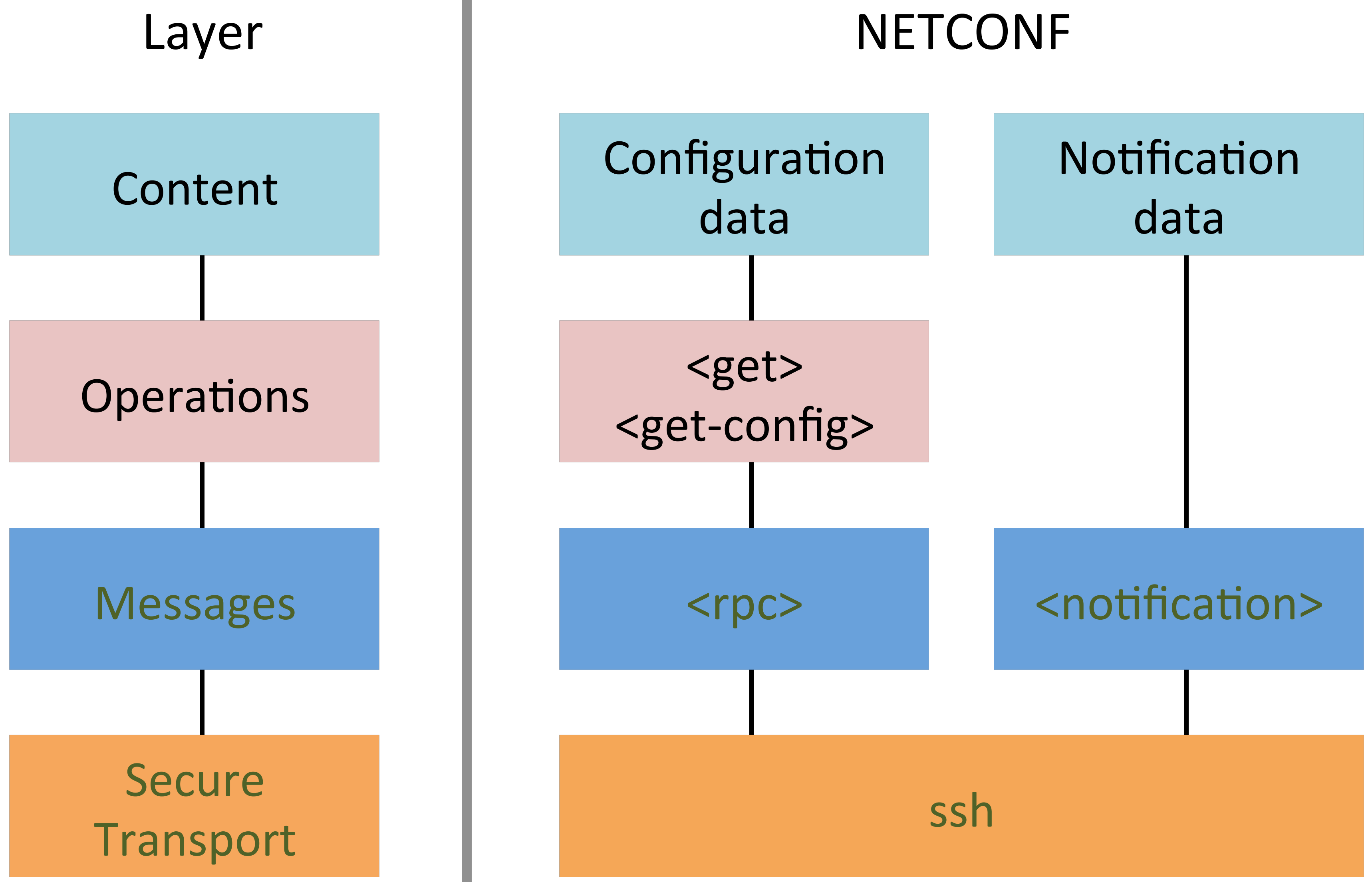
Overview and Objectives

This presentation uses a set of common configuration management tasks to walk through the main features of the NETCONF protocol.

After this presentation, you should be able to:

- Obtain desired configuration attributes from a device using NETCONF
- Configure a network device using NETCONF
- Understand NETCONF transactions

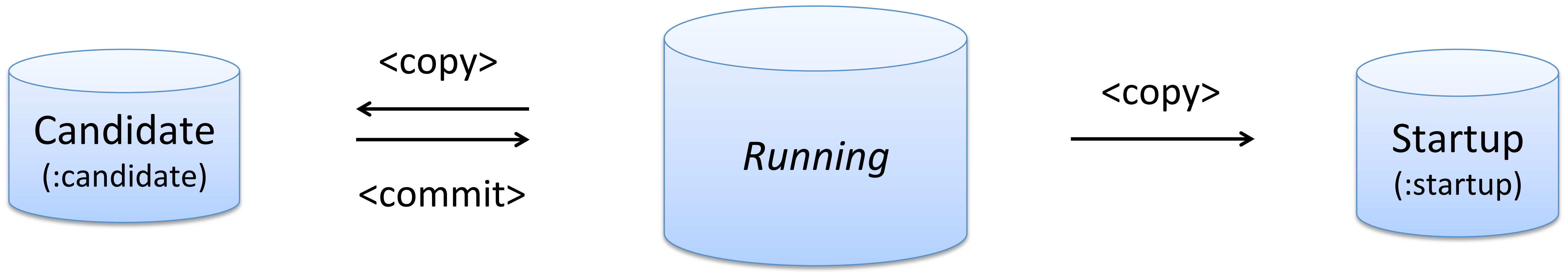
NETCONF Layering Model



Example

```
<rpc
   xmlns="urn:ietf:params:x
   ml:ns:netconf:base:1.0">
    <edit-config>
        <config>
          ...Content...
        </config>
    </edit-config>
</rpc>
```

NETCONF Datastores

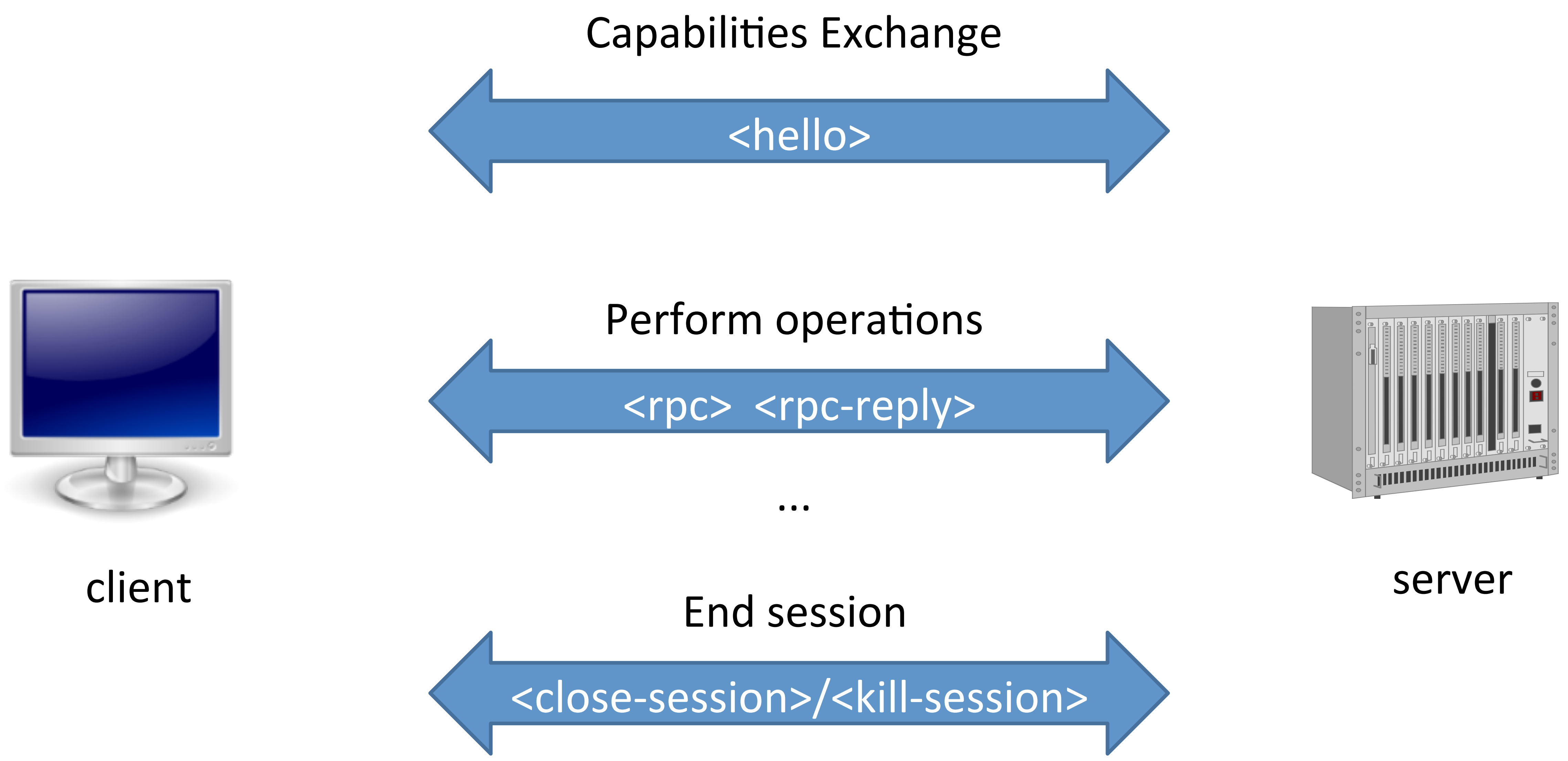


Working copy to manipulate with no impact on current configuration

Complete and active configuration

Configuration loaded by the device at startup

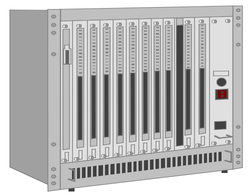
Basic NETCONF Session



Capabilities Exchange - Hello

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
   <capabilities>
      <capability>urn:ietf:params:netconf:base:1.1</capability>
   </capabilities>
</hello>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
   <capabilities>
      <capability>urn:ietf:params:netconf:base:1.1</capability>
      <capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability
      <capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
      ...
   </capabilities>
   <session-id>5</session-id>
</hello>
```



Some Terminology

1. Operation: A specific remote procedure call, as used within the NETCONF protocol
2. Operations have parameters
3. Parameters may have attributes

```
<rpc me 1 ge 2 ="101” xml 3 =”urn:ietf:param
   <get>
      <filter type="subtree">
         <top xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces";
            <interfaces>
            </interfaces>
         </top>
      </filter>
   </get>
</rpc>
```

Getting Data

How do I get all configuration and operational data?

We will use:

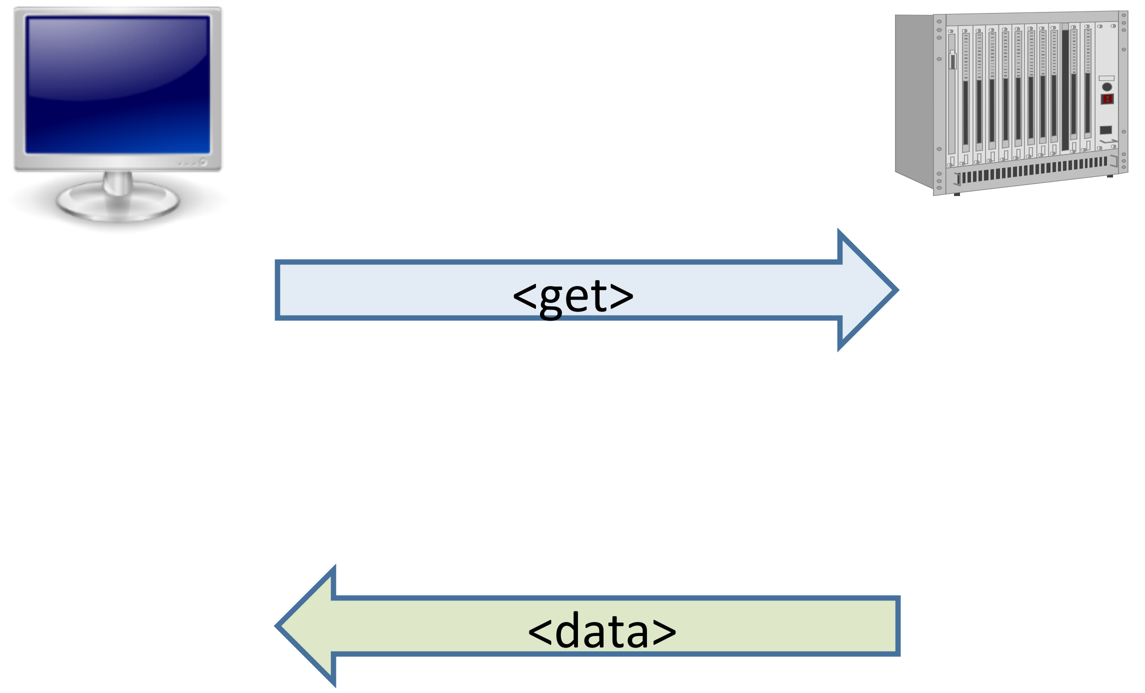
- The `<get>` operation to get the configuration and operational data in a datastore
- The `<get-config>` operation to get only the configuration data in a datastore

Example of using the <get> operation

Obtaining All Data from device

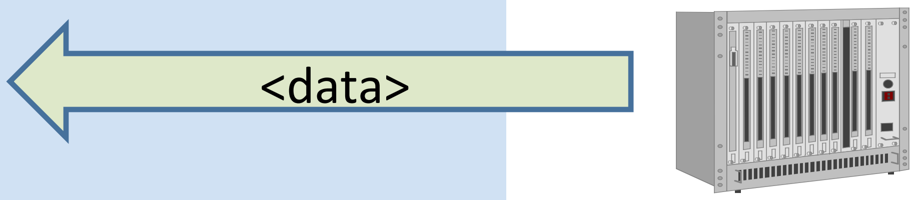
```
<rpc message-id="1"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get/>
</rpc>
```

```
<rpc-reply message-id="1‟
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <data>
         <!-- ... entire set of data returned ... -->
   </data>
</rpc-reply>
```



More Realistic <get> Response

```
<rpc-reply message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>eth0</name>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address>
            <ip>2001:db8:c18:1::3</ip>
            <prefix-length>128</prefix-length>
          </address>
        </ipv6>
      </interface>
      <interface>
        <name>eth1</name>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address>
            <ip>2001:db8:c18:2::1</ip>
            <prefix-length>128</prefix-length>
          </address>
        </ipv6>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```



Filtering Data

How do I filter to get data for just one interface instead of all?

We will use:

- The `<get>` or `<get-config>` operations
- The `<filter>` parameter to select a particular subtree in the reply

Example of Filtering Data

Return just the *interfaces* list

```
<rpc message-id="101” xmlns=”urn:ietf:param
   <get>
      <filter type="subtree">
         <top xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces";
            <interfaces>
            </interfaces>
         </top>
      </filter>
   </get>
</rpc>
```

Return the configuration data for just the *eth0* interface

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:
1.0">
  <get>
    <filter xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interfaces>
        <interface>
          <name>eth0</name>
        </interface>
      </interfaces>
    </filter>
  </get>
</rpc>
```

Reply to a filtered <get> on leaf

```
<rpc-reply message-id="1"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>eth0</name>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-
type">ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address>
            <ip>2001:db8:c18:1::3</ip>
            <prefix-length>128</prefix-length>
          </address>
        </ipv6>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

Manipulating Data

How do I manipulate configuration?

Example: Enabling and configuring the IPv6 address for an interface

We will use:

- The `<edit-config>` operation to edit the datastore content
  - The `<target>` parameter to specify the datastore,
- The `<commit>` operation to commit the candidate datastore content to the running datastore

Using <edit-config>

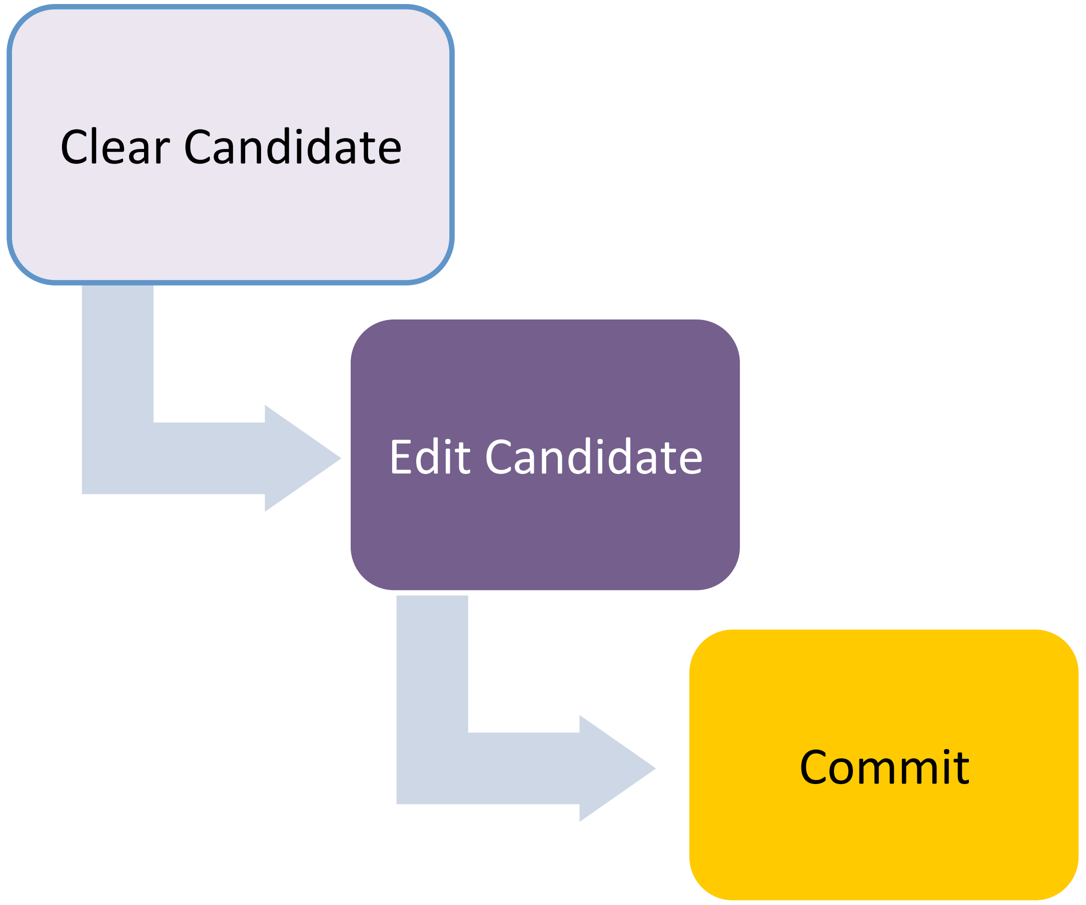
```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"  message-id="1”>
    <edit-config>
        <target>   ...Spcecify the data store to edit ... </target>
        <config> ...  Provide the desired configuration to write ... </config>
    </edit-config>
</rpc>
```

Example: Enabling the Interface

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"  message-id="1">
  <edit-config>
        <target>
            <running/>
        </target>
        <config>
            <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
                <interface>
                    <name>eth0</name>
                    <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-
                              type">ianaift:ethernetCsmacd</type>
                    <enabled>true</enabled>
                </interface>
            </interfaces>
        </config>
  </edit-config>
</rpc>
```

Using <edit-config> on candidate

- Requires :candidate capability



```
<rpc>
   <delete-config>
      <target><candidate/></target>
   </delete-config>
</rpc>
```

```
<rpc>
  <edit-config>
     <target>
       <candidate/>
     </target>
     <config>
          ...New Configuration...
     </config>
  </edit-config>
</rpc>
```

```
<rpc>
  <commit\>
</rpc>
```

Example: Adding IPv6 Address

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <candidate/>
      </target>
      <config>
         <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
            <interface>
               <name>eth0</name>
               <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
                  <address>
                     <ip>2001:db8:c18:1::3</ip>
                     <prefix-length>128</prefix-length>
                  </address>
               </ipv6>
            </interface>
         </interfaces>
      </config>
   </edit-config>
</rpc>
```

...and then commit

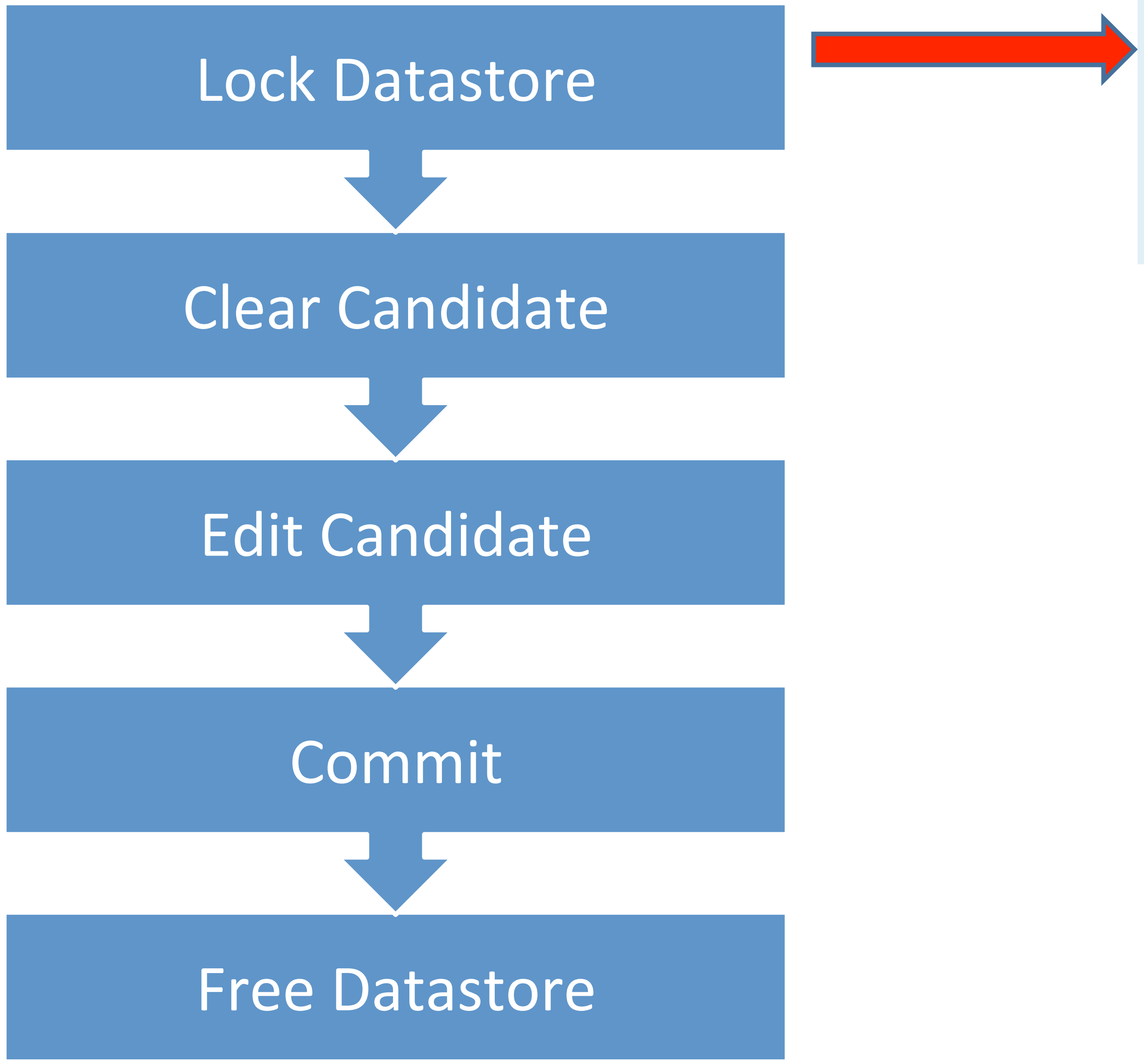
Locking

I don't want others to change the configuration while I'm editing it!

We will use:

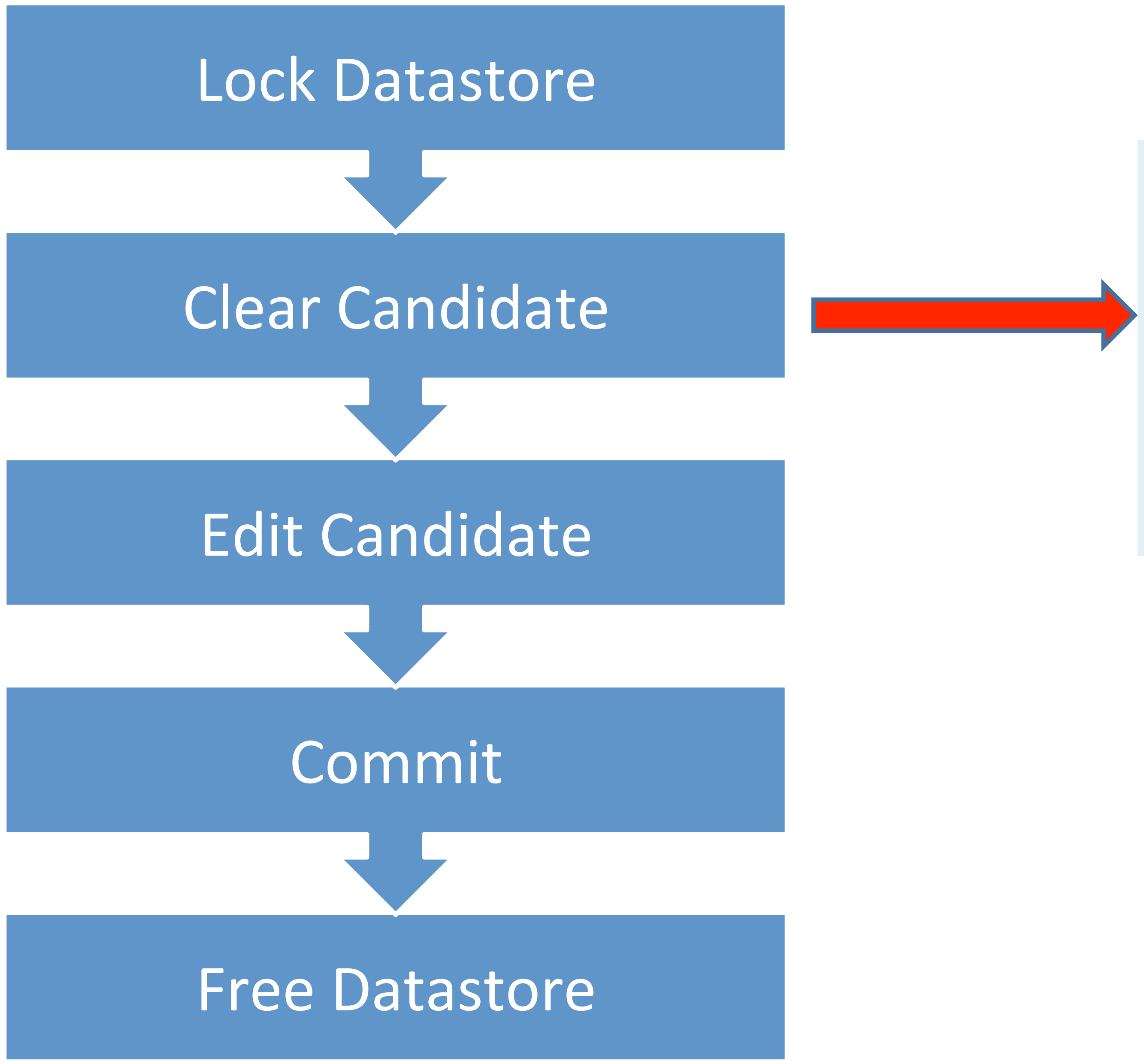
- The `<lock>` operation to lock a datastore
- The `<delete-config>` operation to clear the datastore
- The `<edit-config>` operation to edit the datastore content
- The `<commit>` operation to commit candidate to running
- The `<unlock>` operation to lock a datastore

Locking the Running Datastore



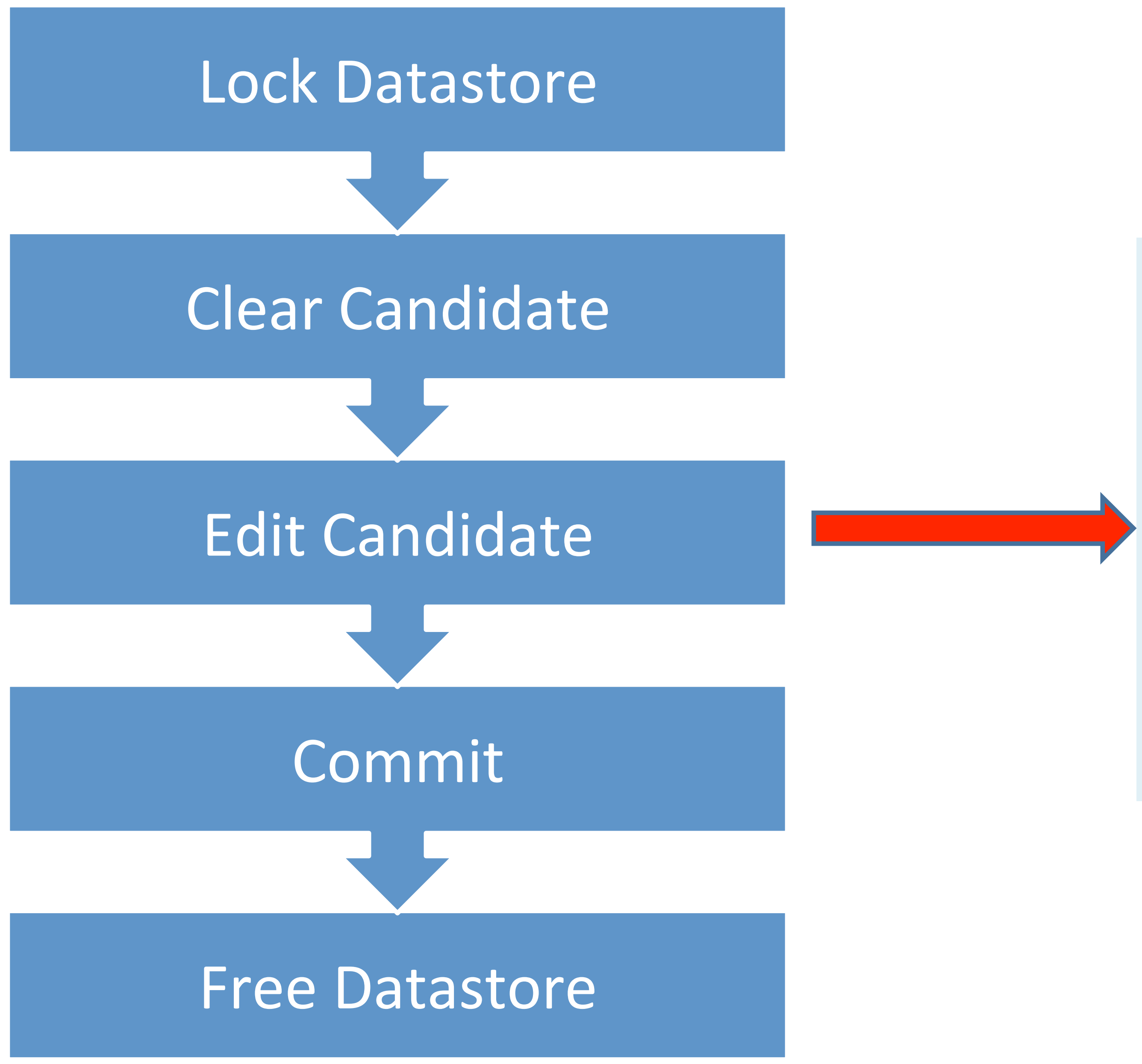
```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="1">
  <lock>
    <target><running/></target>
  </lock>
</rpc>
```

Clear the Candidate Datastore



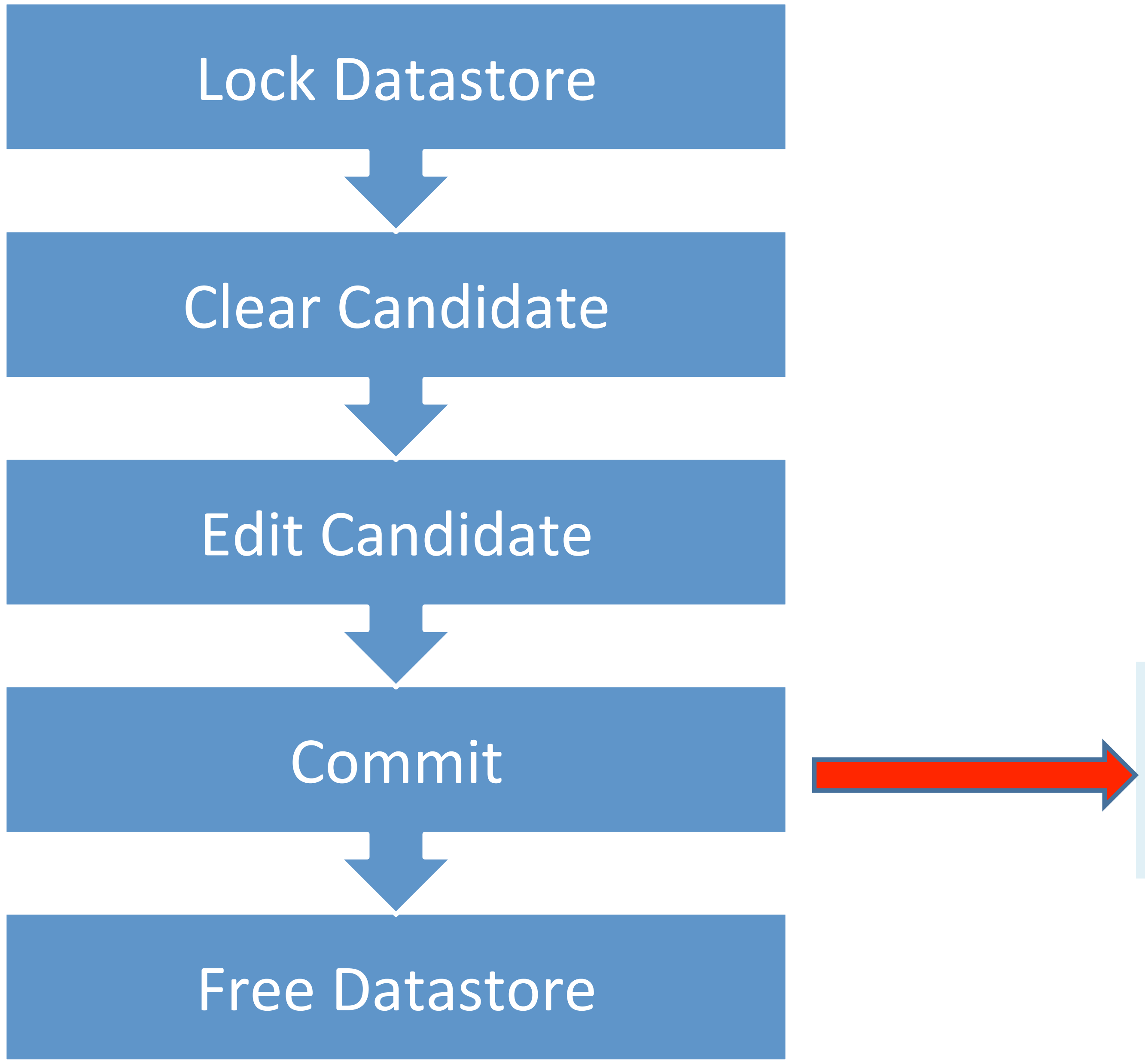
```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="3">
  <delete-config>
    <target>
      <candidate/>
    </target>
  </delete-config>
</rpc>
```

Edit the Candidate Datastore



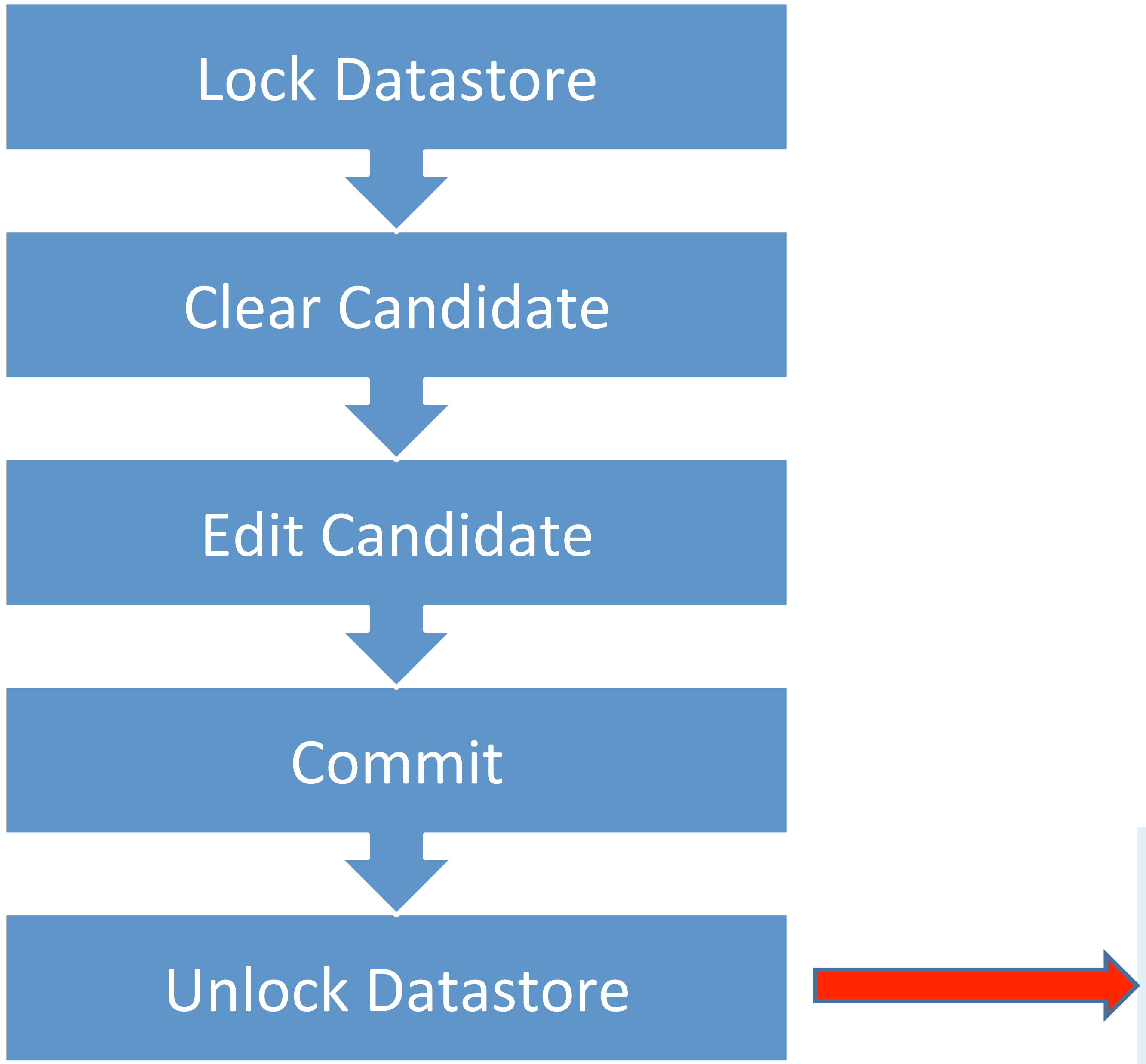
```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="4">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
        . . . Configuration data . . .
    </config>
  </edit-config>
</rpc>
```

Commit the Candidate to the Running



```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="5">
  <commit/>
</rpc>
```

Unlock the Running Datastore



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

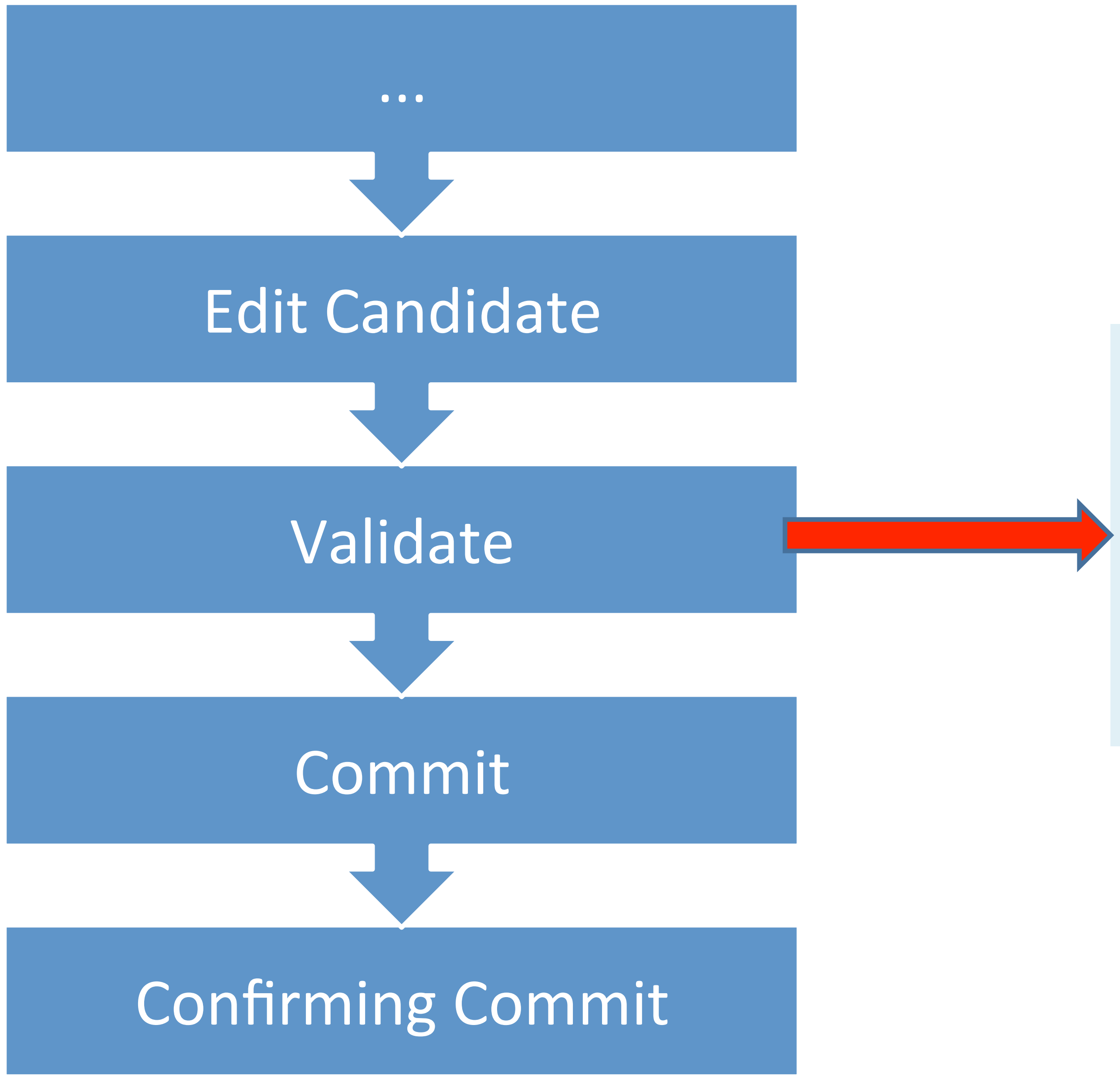
Validation and Rollback

I want to test the configuration before I commit and cancel out if necessary!

We will use:

- The `<validate>` operation to validate the content of a datastore
- The `<commit>` operation to commit candidate to running
  - The `<confirmed>` parameter to denote a confirmed commit
  - The `<persist>` parameter to specify a commit identifier
  - The `<confirm-timeout>` parameter to specify a timeout before rollback

Validation

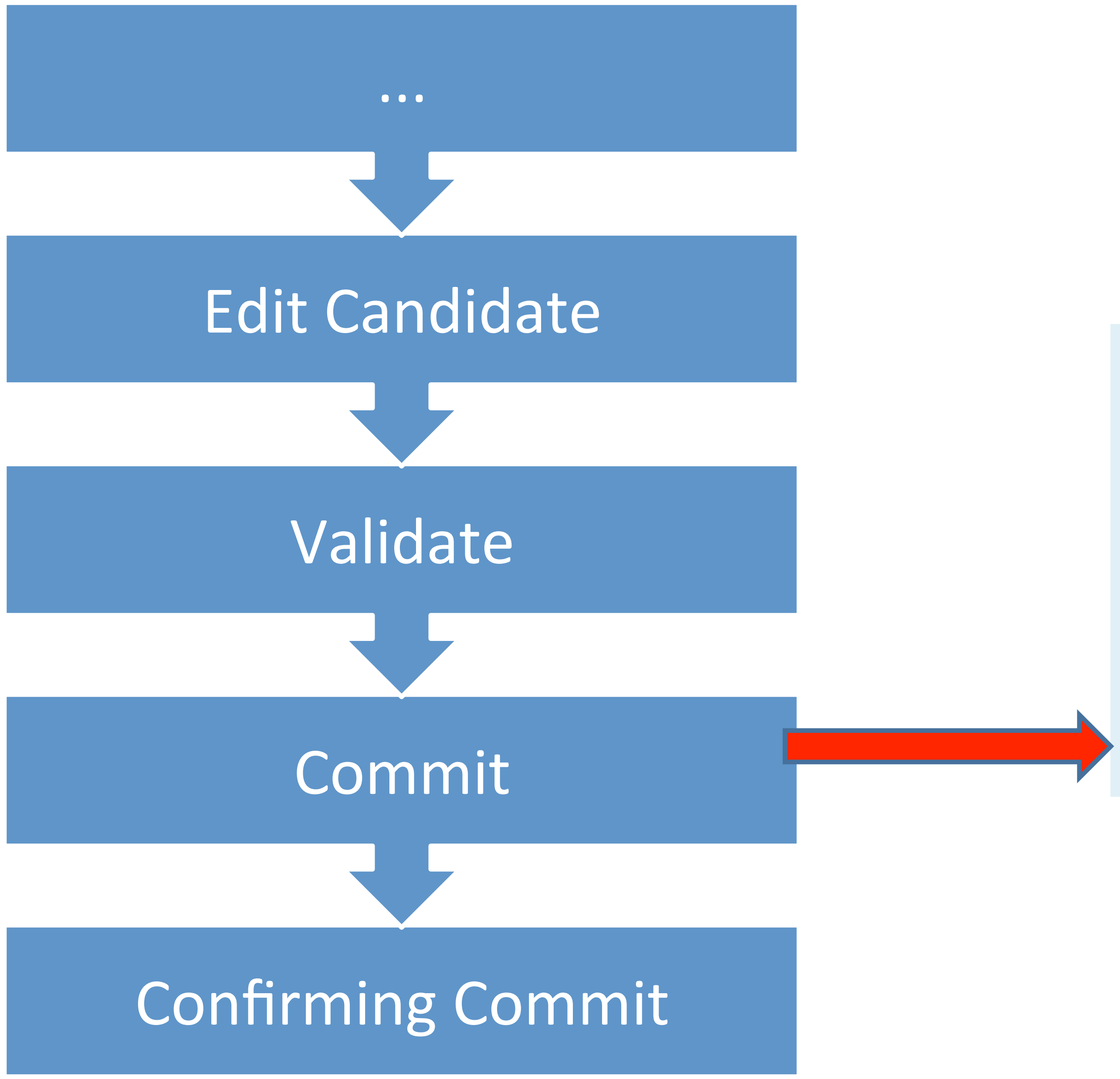


Check for syntactical and semantic errors.

```
<rpc message-id="5"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <validate>
    <source>
      <candidate/>
    </source>
  </validate>
</rpc>
```

If ok is received back proceed to Commit

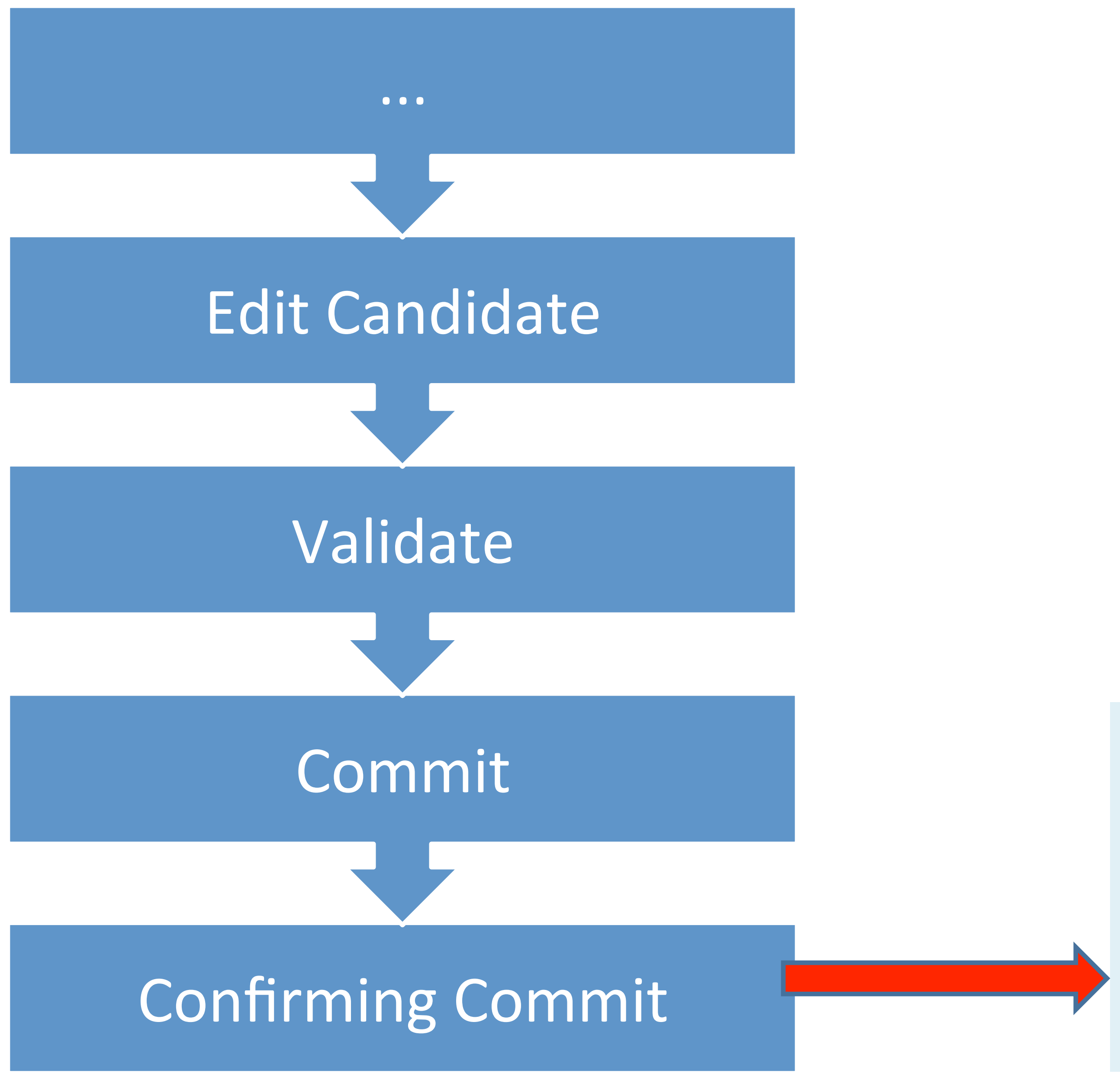
Confirmed Commit



- Requires :confirmed-commit capability
- Commit for 10 seconds then timeout and revert if confirmation not received

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="6"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" >
  <commit>
    <confirmed/>
    <confirm-timeout>10</confirm-timeout>
    <persist>IQ,d4668</persist>
  </commit>
</rpc>
```

Confirming Commit



```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="7"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" >
  <commit>
    <persist>IQ,d4668</persist>
  </commit>
</rpc>
```

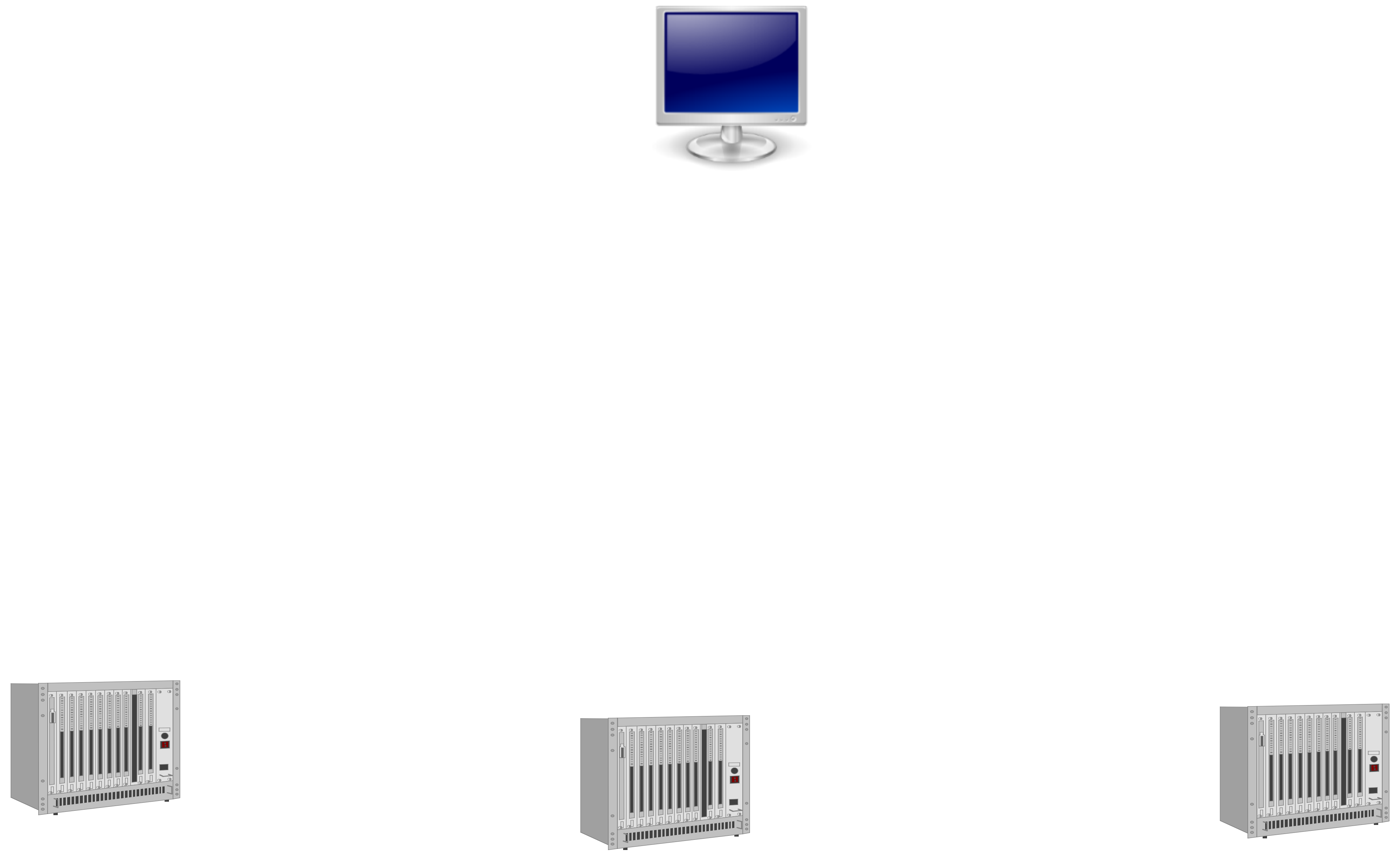
Configuring Multiple Devices

I want to configure multiple devices at once and rollback if anyone fails

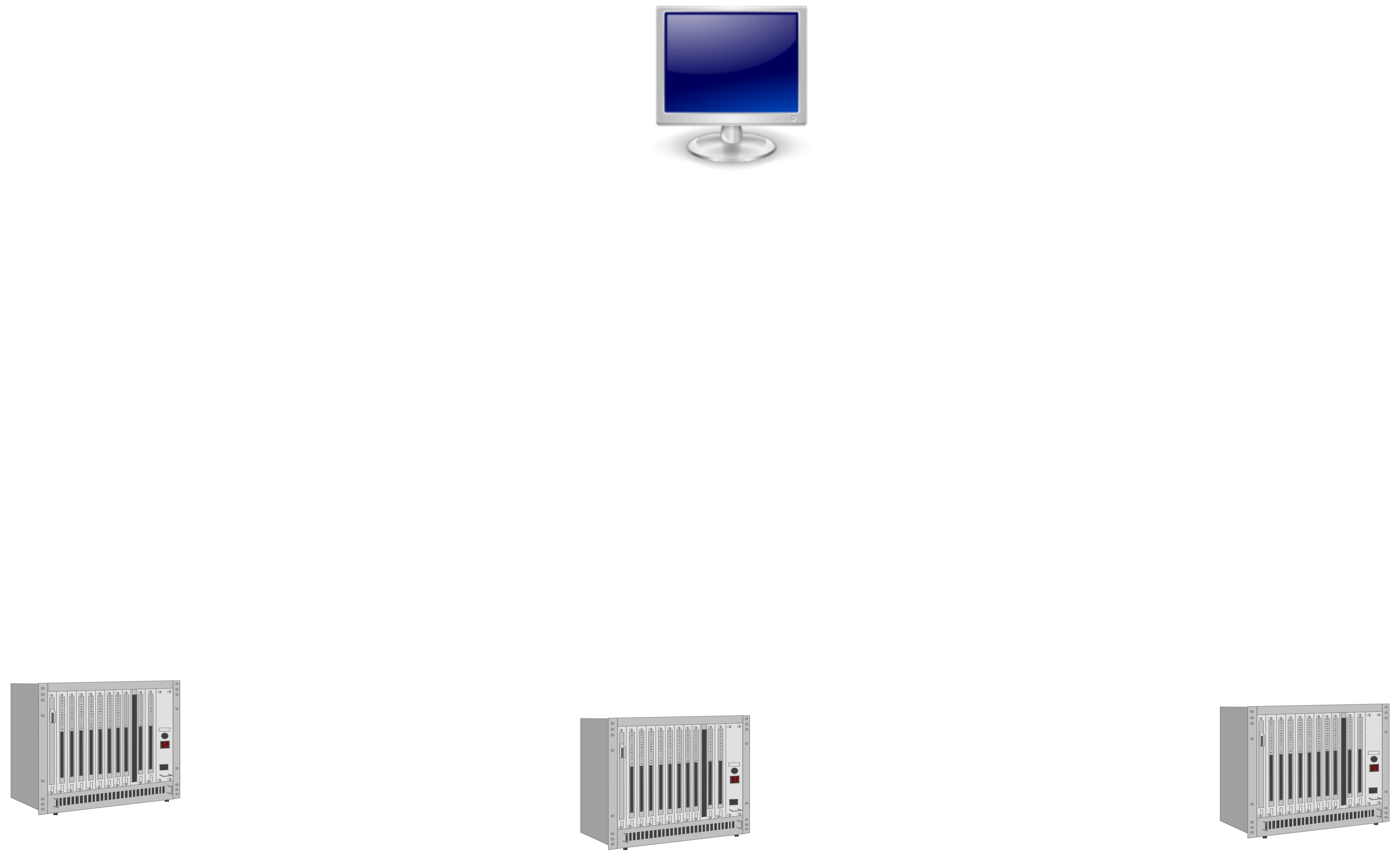
This leverages a combination of parallel sessions and confirmed commits. We will use the same steps as in the previous example, but towards three network devices.

This allows for two-phase commit transactions

Step #1: Prepare



Step #1: Commit



Summary

You should now be able to:

- Obtain desired configuration attributes from a device using NETCONF
- Configure a network device using NETCONF
- Understand NETCONF transactions

Back Matter

- This material was originally developed by Charlie Justus and Carl Moberg with the support of Cisco Systems, special thanks to:
  - Kevin Serveau

Changelog

- 1.0 (2015-10-05) – Initial version
  Carl Moberg <camoberg@cisco.com>