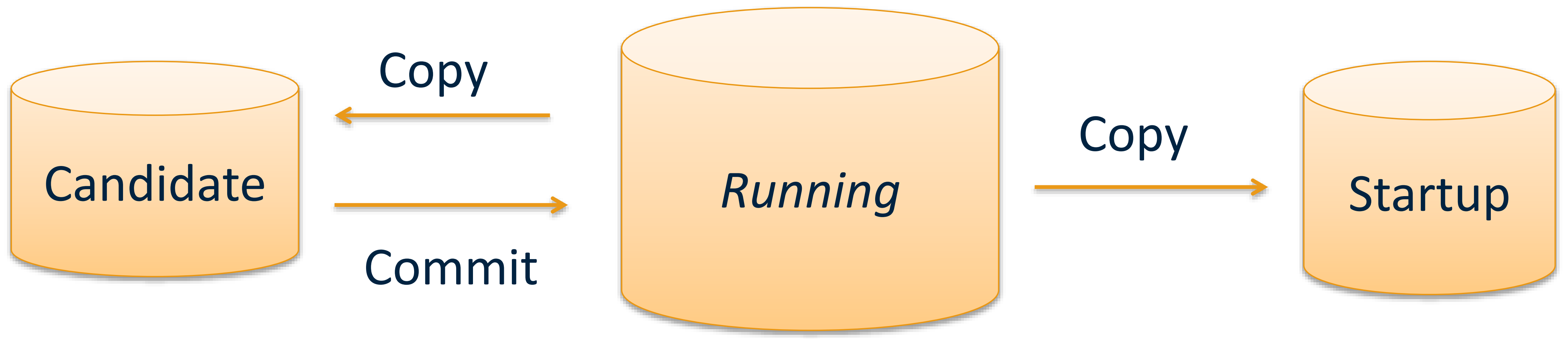
NETCONF Conceptual Databases



The optional **Candidate Datastore** represents a working copy for manipulating configuration data with no impact on current configuration

The mandatory **Running Datastore** represents the complete and active configuration on the network device

The optional **Startup Datastore** is loaded by the device when it boots.

NETCONF Capabilities

- A capability is a set of functionality that supplements base NETCONF spec
- Capabilities augment:
  - Additional operations
  - Content allowed inside these operations
- Capabilities advertised by server during session establishment

- Base NETCONF specification provides very restricted set of operations for lightweight server implementations

python

```
with manager.connect(host=host, port=22, username=user) as m:
    assert(":validate" in m.server_capabilities)
    m.edit_config(target='running', config=snippet,
                  test_option='test-then-set')
```

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.1</capability>
    <capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
    <capability>urn:ietf:params:netconf:capability:confirmed-commit:1.1</capability>
    <capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
    <capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
    <capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
    <capability>urn:ietf:params:xml:ns:yang:ietf-inet-types?module=ietf-inet-types&amp;revision=2013-07-15</capability>
    <capability>urn:ietf:params:xml:ns:yang:ietf-netconf-acm?module=ietf-netconf-acm&amp;revision=2012-02-22</capability>
    <capability>urn:ietf:params:xml:ns:yang:ietf-yang-types?module=ietf-yang-types&amp;revision=2013-07-15</capability>
  </capabilities>
  <session-id>14</session-id>
</hello>
```

Common Operations

Data Manipulation
- `<get>`
- `<get-config>`
- `<edit-config>`
- `<copy-config>`
- `<delete-config>`
- `<discard-changes>` *(:candidate)*

Session Management
- `<close-session>`
- `<kill-session>`

Locking
- `<lock>`
- `<unlock>`

Transaction Management
- `<commit>` *(:candidate, :confirmed)*
- `<cancel-commit>` *(:candidate)*

Schema Management
- `<get-schema>` *(:monitoring)*

RPC Extensions
- `<rpc>`

Anatomy of NETCONF Sessions

Ambitious version:

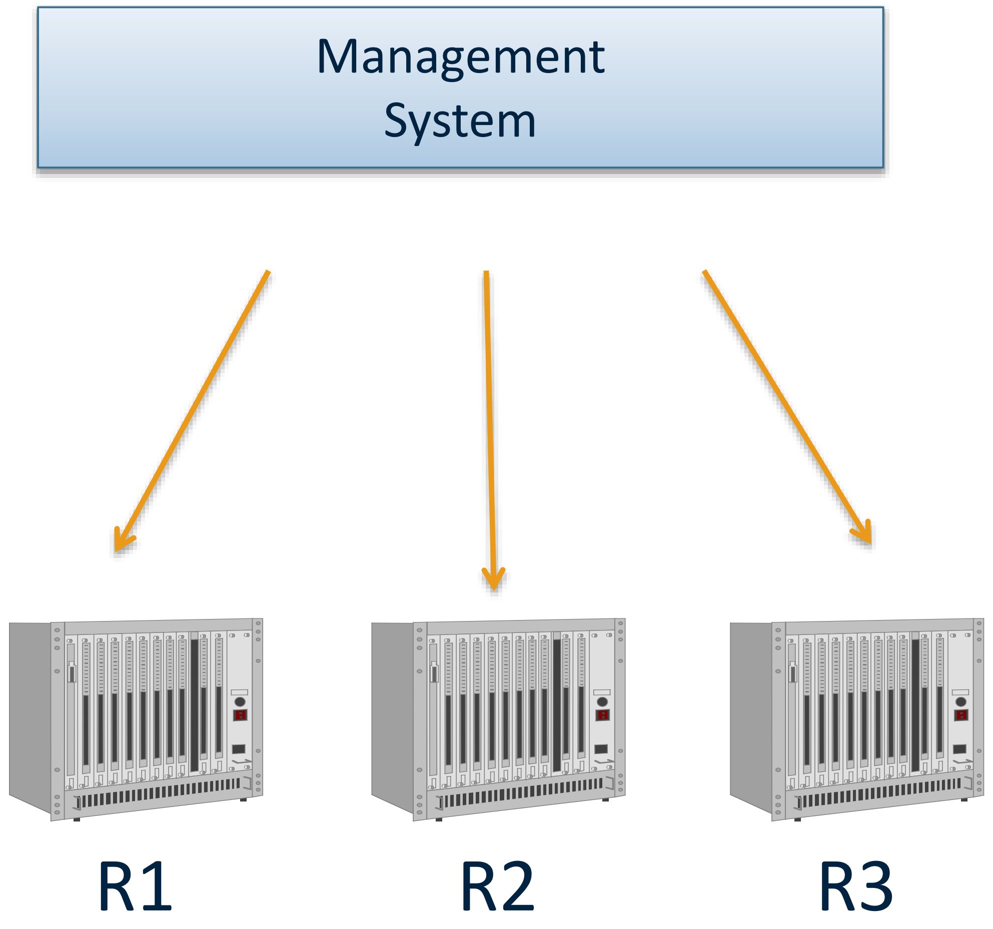
- **Hello** exchange including capabilities
- **Lock** running
- **Lock** candidate
- **Discard** changes on candidate
- **Edit** config on candidate
- **Commit** confirmed (with timeout)
- **Confirm** commit
- **Copy** running to startup
- **Unlock** candidate
- **Unlock** running

Short version:

- **Hello** exchange including capabilities
- **Edit** config on running database

```
with manager.connect(host=host, port=22, username=user) as m:
    if(":candidate" in m.server_capabilities):
        with m.locked(target='candidate'):
            m.discard_changes()
            ...
    else:
        m.edit_config(target='running', config=cfg)
```

Distributed Transactions (for Bonus Points)

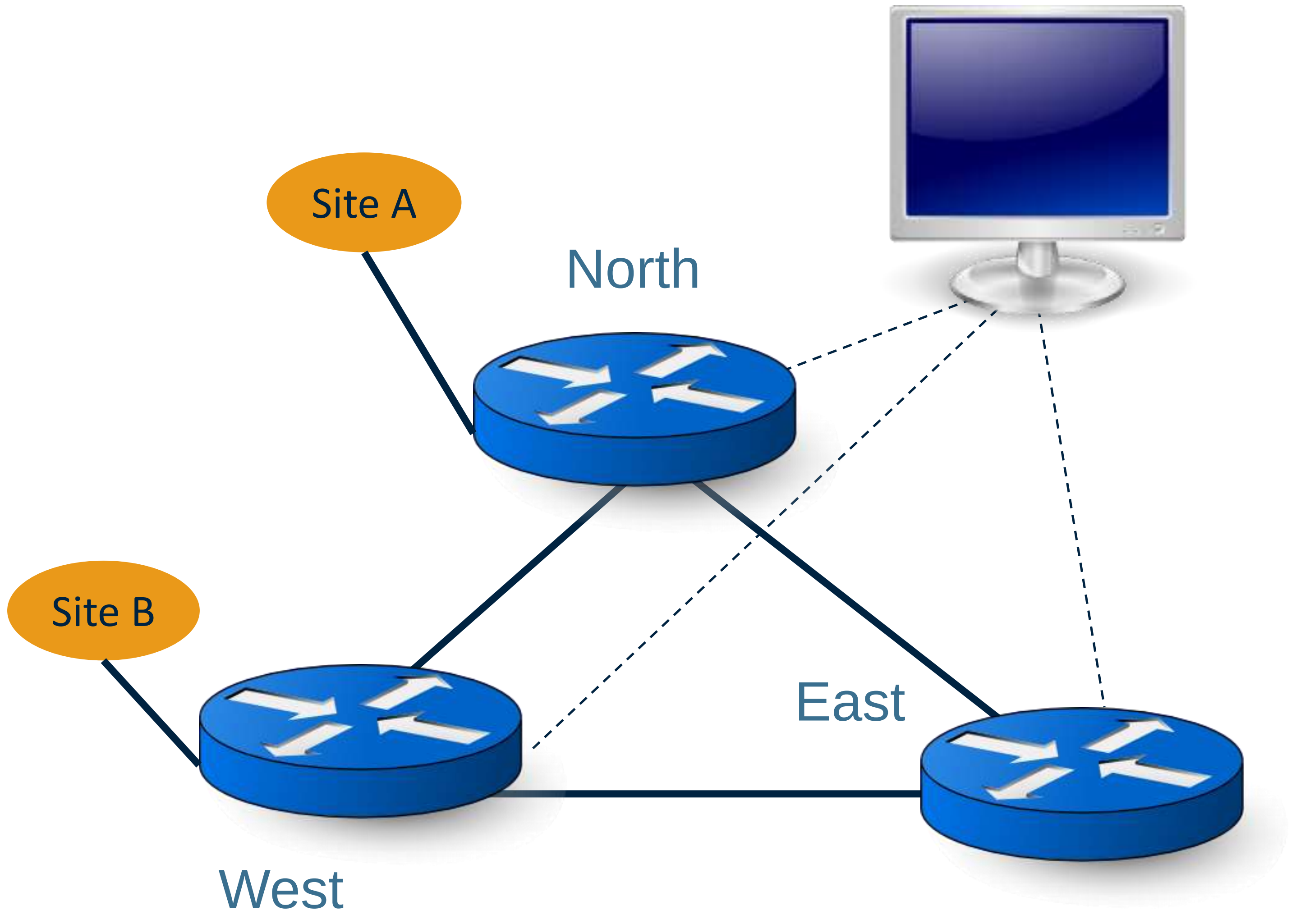


1. Connect to and lock R1, R2, R3
2. Edit candidate databases and commit with timeout
3. (Optionally) do assurance checks during timeout
4. Confirm commit, copy to startup and release locks

Transaction context manager simply kills all sessions on communication failure, failed commits -> Rollback

Example: VPN provisioning using Transactions

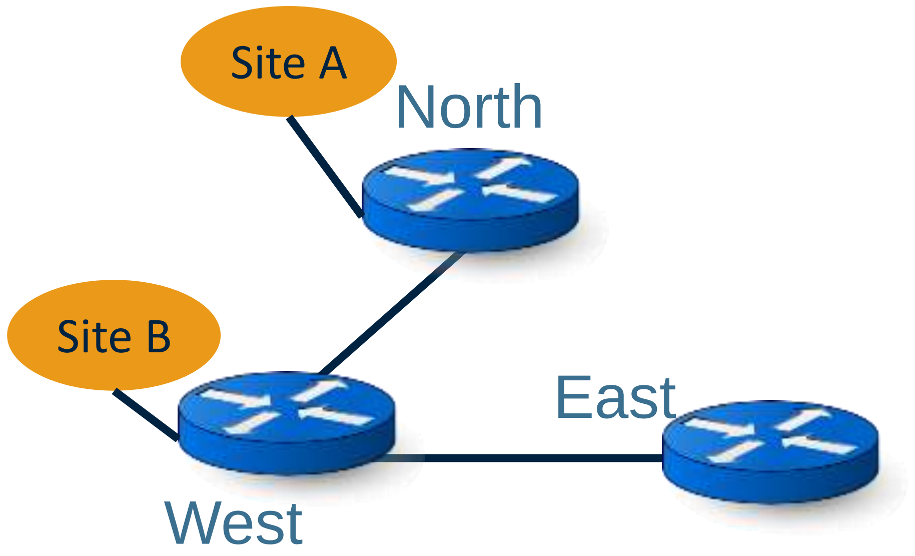
VPN Scenario



- An operator owns a network of routers and other equipment from different vendors
- They have a management station connected to all the devices in the network to provision and monitor services
- Now, we need to set up a VPN between two customer sites
- There is no point what so ever to make any changes on any device unless all changes succeed
- We need a Network-wide Transaction!

Remember: This is a model and protocol tutorial!

Hello



```
>>>>> Router-West (Sun Nov 15 14:41:25 CET 2009)
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.1</capability>
  </capabilities>
</hello>
```

```
<<<< Router-West (Sun Nov 15 14:41:25 CET 2009)
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <capabilities>
     <capability>urn:ietf:params:netconf:base:1.1</capability>
     <capability>http://tail-f.com/ns/netconf/actions/1.0</capability>
     <capability>http://tail-f.com/ns/aaa/1.1</capability>
     <capability>http://tail-f.com/ns/example/quagga/1.0</capability>
  </capabilities>
  <session-id>4</session-id>
</hello>
```

Lock the running data stores

```
>>>>> Router-West (Sun Nov 15 15:24:33 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="2">
  <nc:lock>
     <nc:target>
       <nc:candidate></nc:candidate>
     </nc:target>
  </nc:lock>
</nc:rpc>
```

```
<<<< Router-West (Sun Nov 15 15:24:33 CET 2009)
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="2">
   <ok></ok>
</rpc-reply>
```

Lock the running data stores

```
>>>>> Router-West (Sun Nov 15 15:24:33 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="3">
  <nc:lock>
     <nc:target>
       <nc:running></nc:running>
     </nc:target>
  </nc:lock>
</nc:rpc>
```

```
<<<< Router-West (Sun Nov 15 15:24:33 CET 2009)
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="3">
   <ok></ok>
</rpc-reply>
```

Clear the candidate data stores

```
>>>>> Router-West (Sun Nov 15 15:24:32 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="1">
   <nc:discard-changes></nc:discard-changes>
</nc:rpc>
```

```
<<<< Router-West (Sun Nov 15 15:24:33 CET 2009)
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="1">
   <ok></ok>
</rpc-reply>
```

Edit the candidates

```
>>>>> Router-West (Sun Nov 15 15:24:33 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="5">
  <nc:edit-config>
     <nc:target><nc:candidate></nc:candidate></nc:target>
     <nc:config>
       <quagga:system xmlns:quagga="http://tail-f.com/ns/example/quagga
       <quagga:vpn>
         <quagga:ipsec>
           <quagga:tunnel>
             <quagga:name>volvo-0</quagga:name>
             <quagga:local-endpoint>10.7.7.4</quagga:local-endpoint>
             <quagga:local-net>33.44.55.0</quagga:local-net>
             <quagga:local-net-mask>255.255.255.0</quagga:local-net-mas
             <quagga:remote-endpoint>10.3.4.1</quagga:remote-endpoint>
             <quagga:remote-net>62.34.65.0</quagga:remote-net>
             <quagga:pre-shared-key>ford</quagga:pre-
             <quagga:encryption-algo>default
             [...]
```

Validate candidates

```
>>>>> Router-West (Sun Nov 15 15:24:33 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="6">
  <nc:validate>
    <nc:source>
      <nc:candidate></nc:candidate>
    </nc:source>
  </nc:validate>
</nc:rpc>
```

```
<<<< Router-West (Sun Nov 15 15:24:33 CET 2009)
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="6">
   <ok></ok>
</rpc-reply>
```

Commit candidates to running

```
>>>>> Router-West (Sun Nov 15 15:24:33 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="7">
   <nc:commit></nc:commit>
</nc:rpc>
```

```
<<<< Router-West (Sun Nov 15 15:24:33 CET 2009)
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="7">
   <ok></ok>
</rpc-reply>
```

Unlock candidates

```
>>>>> Router-West (Sun Nov 15 15:24:33 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="8">
  <nc:unlock>
    <nc:target>
      <nc:candidate></nc:candidate>
    </nc:target>
  </nc:unlock>
</nc:rpc>
```

```
<<<< Router-West (Sun Nov 15 15:24:33 CET 2009)
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="7">
   <ok></ok>
</rpc-reply>
```

Using confirmed-commit

- Now do the same thing again, but instead of commit:

```
>>>>> Router-West (Sun Nov 15 15:29:19 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="16">
   <nc:commit>
           <nc:confirmed></nc:confirmed>
           <nc:confirm-timeout>120</nc:confirm-timeout>
   </nc:commit>
</nc:rpc>
```

```
<<<< Router-West (Sun Nov 15 15:29:19 CET 2009)
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="16">
   <ok></ok>
</rpc-reply>
```

Disaster happens

- One of the devices disconnected
- The management station disconnects all the rest
- They all roll back to the previous configuration
- The management station reconnects

```
>>>>> Router-West (Sun Nov 15 15:30:22 CET 2009)
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.1</capability>
  </capabilities>
</hello>
```

```
<<<< Router-West (Sun Nov 15 15:30:22 CET 2009)
<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.
```

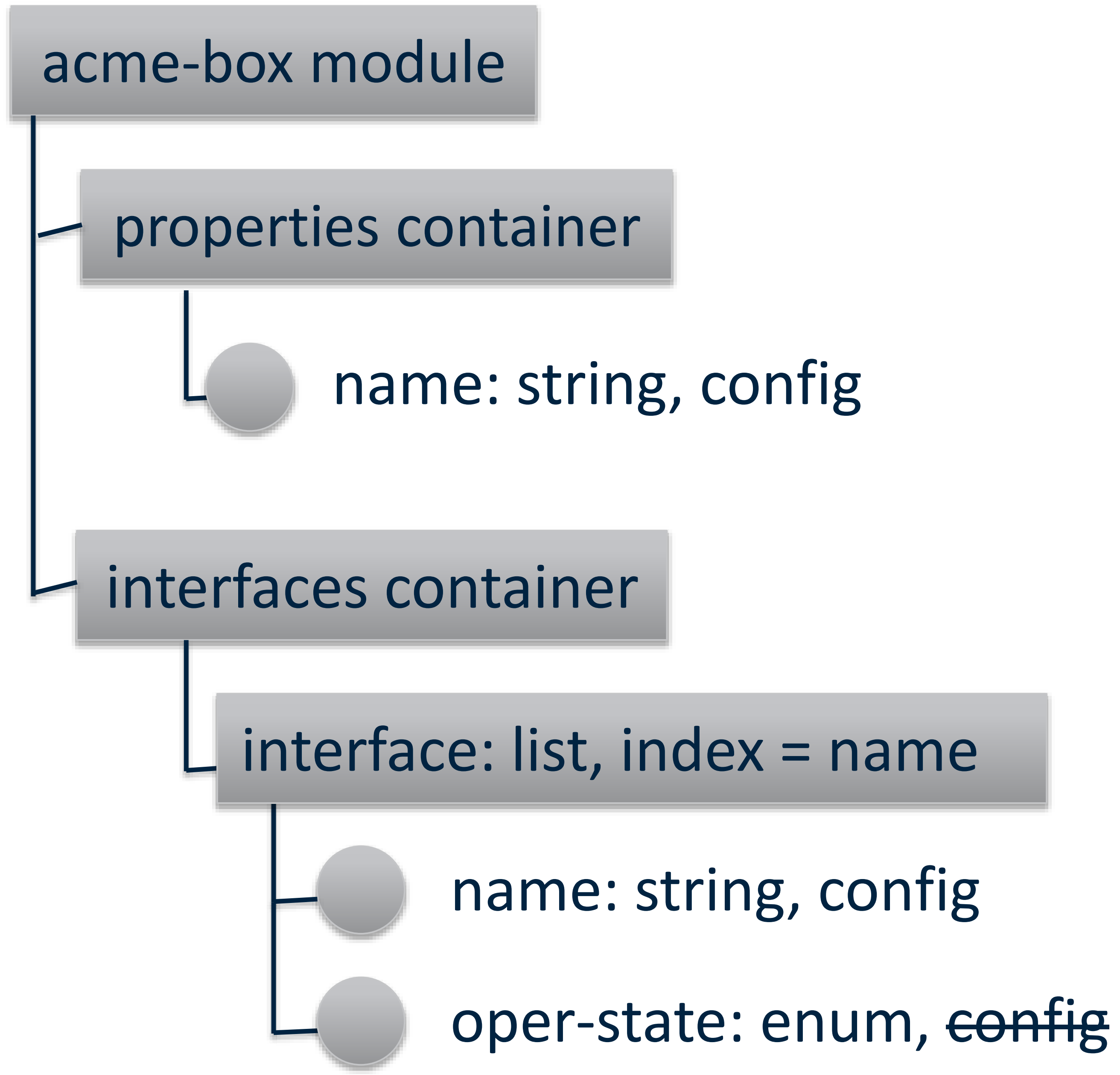
Common NETCONF Use Cases

- Network-wide transactions
- Applying and testing a configuration
- Testing and rejecting a configuration
- Rollback when device goes down
- Transactions requiring all devices to be up
- Backlogging transactions
- Synchronizing

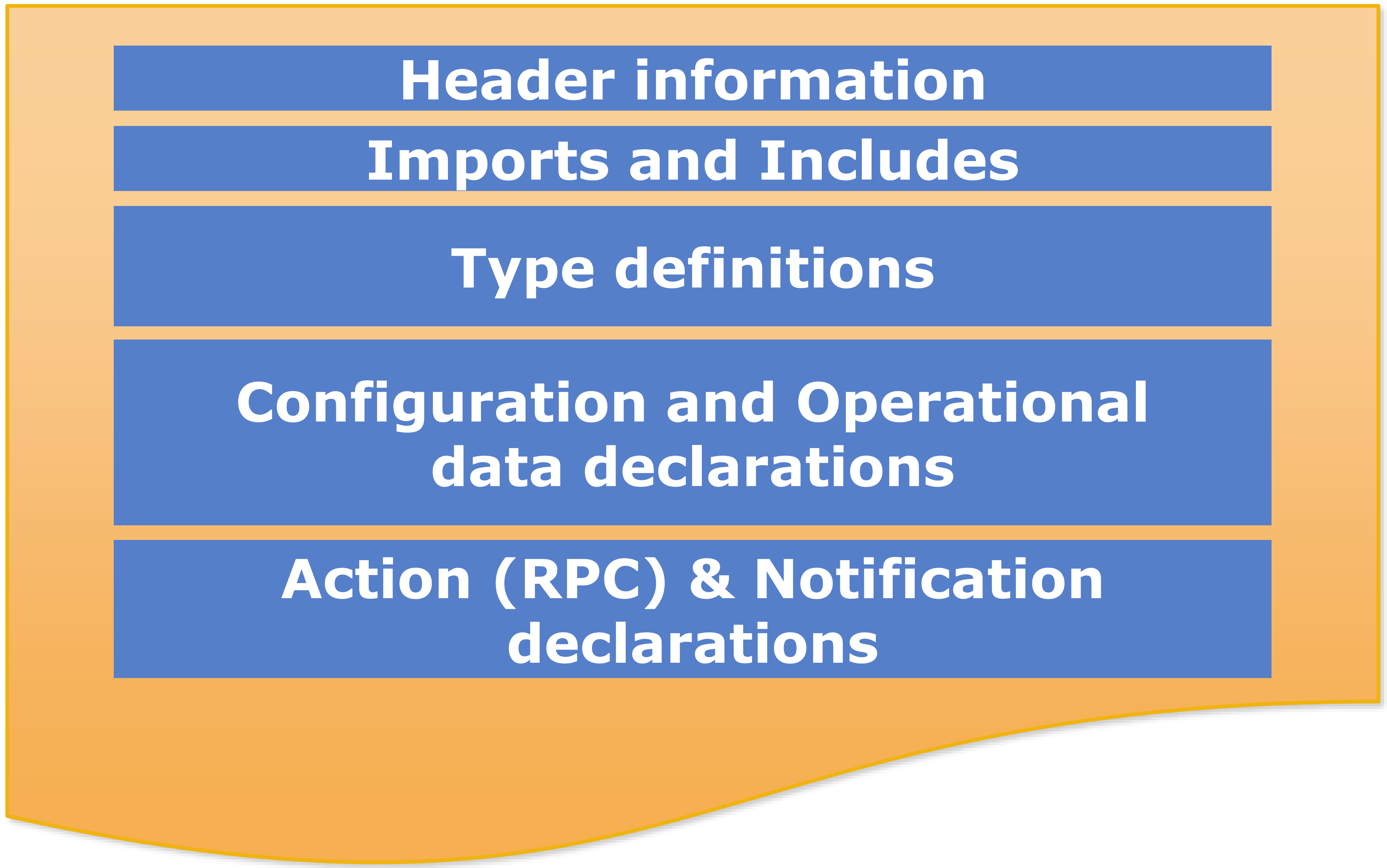
YANG Deeper Dive

YANG ?

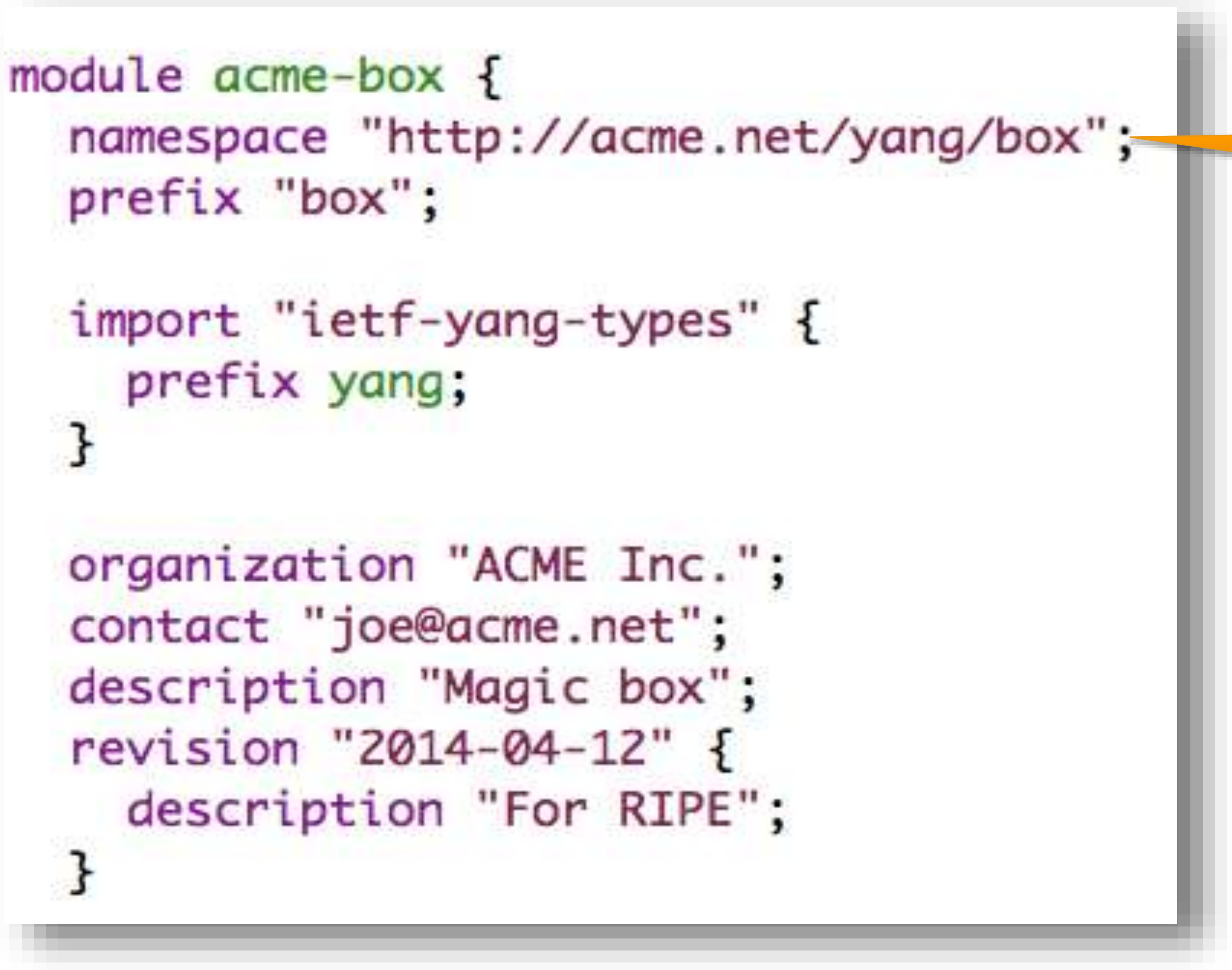
- Data modeling language
  - Configuration data
  - State data
- Tree structured
- Close to device instrumentation
- Managing device features
- Data and Types
- Constraints
- Augmentation
- Reusable structures
- Extensions
- SMI translation
- XML and JSON
  - NETCONF Transport Encoding
  - RESTCONF Transport Encoding



YANG Module Contents


Header information
Imports and Includes
Type definitions
Configuration and Operational data declarations
Action (RPC) & Notification declarations

YANG Header



```
module acme-box {
  namespace "http://acme.net/yang/box";
  prefix "box";

  import "ietf-yang-types" {
    prefix yang;
  }

  organization "ACME Inc.";
  contact "joe@acme.net";
  description "Magic box";
  revision "2014-04-12" {
    description "For RIPE";
  }
```

URI

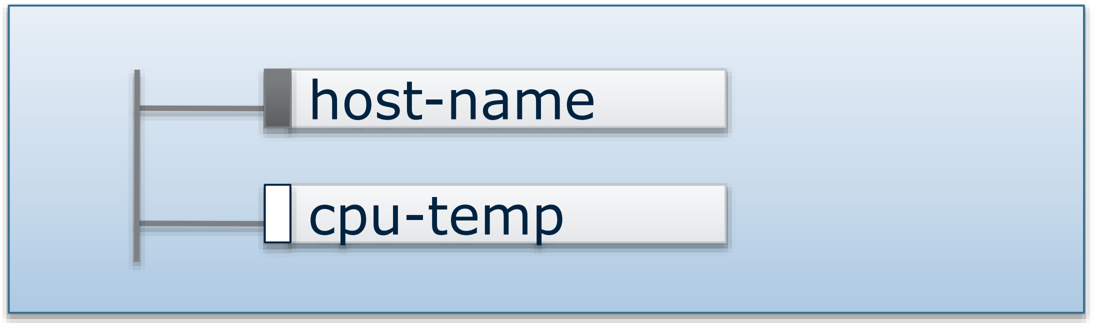
YANG Data Definitions

Leaf Statement

Holds a single value of a particular type

Has no children

```
leaf host-name {
  type string;
  mandatory true;
  config true;
  description "Hostname for this system";
}
leaf cpu-temp {
  type int32;
  units degrees-celsius;
  config false;
  description ”Current temperature in CPU";
}
```



NETCONF XML:

```
<host-name>my.example.com</host-name>
```

cpu-temp **not** returned in NETCONF get-config

Attributes for leaf

`config`	Whether this leaf is a configurable value ("true") or operational value ("false"). Inherited from parent container if not specified
`default`	Specifies default value for this leaf. Implies that leaf is optional
`mandatory`	Whether the leaf is mandatory ("true") or optional ("false")
`must`	XPath constraint that will be enforced for this leaf
`type`	The data type (and range etc) of this leaf
`when`	Conditional leaf, only present if XPath expression is true
`description`	Human readable definition and help text for this leaf
`reference`	Human readable reference to some other element or spec
`units`	Human readable unit specification (e.g. Hz, MB/s, °F)
`status`	Whether this leaf is "current", "deprecated" or "obsolete"