



Tutorial: YANG



YANG Module Contents

Header information

Imports & Includes

Type definitions

**Configuration & 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

Data Modeling Nodes

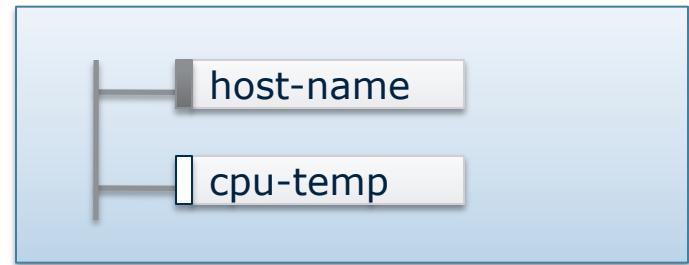
- Leaf
- Leaf-List
- Container
- List

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

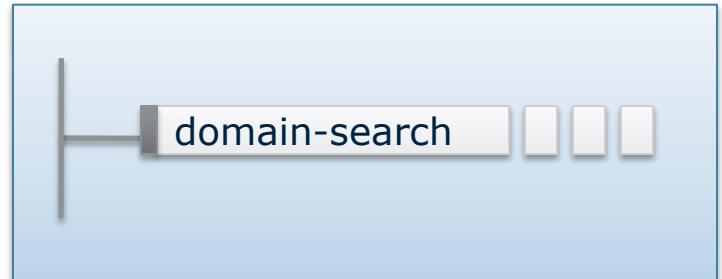
Attributes for leaf

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"

Leaf-list Statement

Holds multiple values of a particular type

Has no children



```
leaf-list domain-search {  
    type string;  
    ordered-by user;  
    description "List of domain names to search";  
}
```

NETCONF operations to insert
first, last, before, after

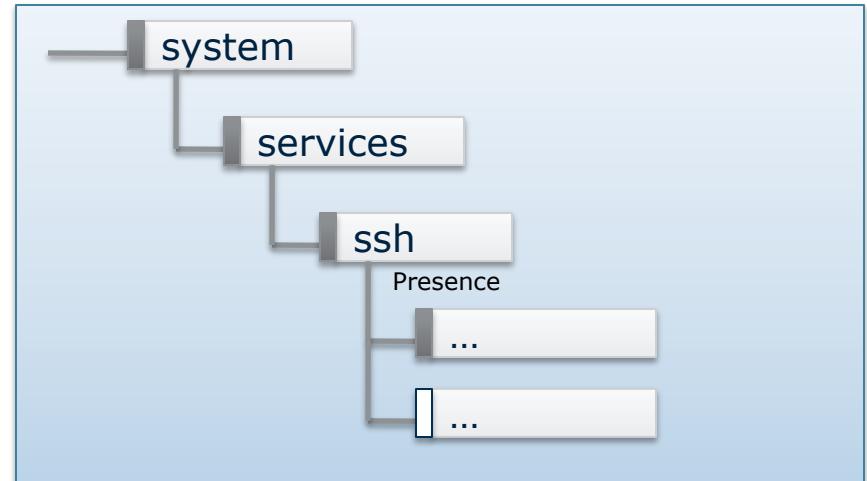
NETCONF XML:

```
<domain-search>high.example.com</domain-search>  
<domain-search>low.example.com</domain-search>
```

Container Statement

Groups related leafs and containers

```
container system {
    container services {
        container ssh {
            presence "Enables SSH";
            description "SSH service specific configuration";
            // more leafs, containers and other things here...
        }
    }
}
```

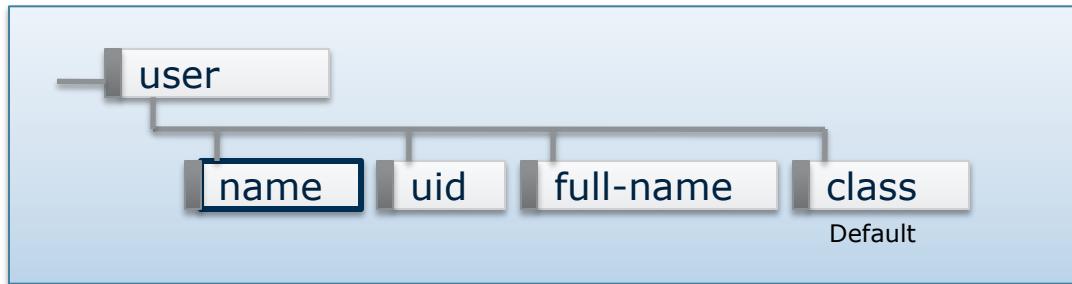


Presence containers explicitly created/deleted by NETCONF client. They also represent config “themselves”. “Normal” containers have no meaning, just organization of data.

NETCONF XML:

```
<system>
  <services>
    <ssh>
    </ssh>
  </services>
</system>
```

List Statement



```

list user {
  key name;
  leaf name {
    type string;
  }
  leaf uid {
    type uint32;
  }
}
leaf full-name {
  type string;
}
leaf class {
  type string;
  default viewer;
}
  
```

NETCONF XML:

```

<user>
  <name>glocks</name>
  ...
</user>
<user>
  <name>snowey</name>
  ...
</user>
  
```

Non-config lists can skip key
Given at create!

NETCONF operations to insert
first, last, before, after

Putting things together

```
module acme-system {
    namespace "http://acme.example.com/system";
    prefix "acme";

    organization "ACME Inc.";
    contact "joe@acme.example.com";
    description
        "The module for entities implementing the
         system interface.

    revision 2007-06-09 {
        description "Initial revision.";
    }
}
```

```
container system {
    leaf host-name {
        type string;
        description "Hostname for this system";
    }

    leaf-list domain-search {
        type string;
        description "List of domain names to search";
    }

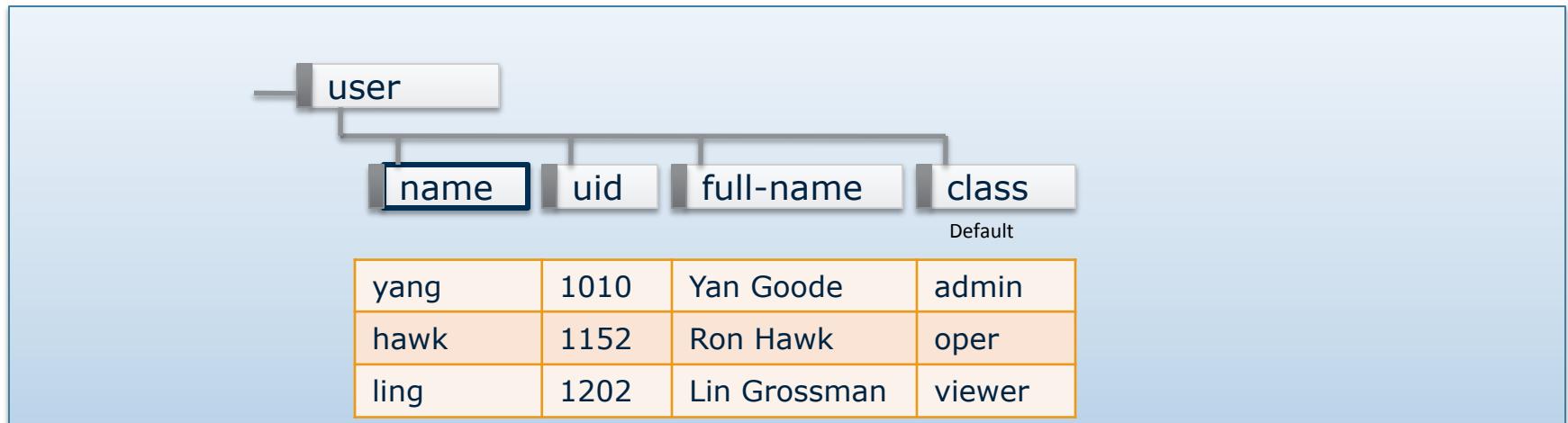
    container login {
        leaf message {
            type string;
            description
                "Message given at start of login session";
        }

        list user {
            key "name";
            leaf name {
                type string;
            }
            leaf full-name {
                type string;
            }
            leaf class {
                type string;
            }
        }
    }
}
```

Attributes for list and leaf-list

max-elements	Max number of elements in list. If max-elements is not specified, there is no upper limit, i.e. "unbounded"
min-elements	Min number of elements in list. If min-elements is not specified, there is no lower limit, i.e. 0
ordered-by	List entries are sorted by "system" or "user". System means elements are sorted in a natural order (numerically, alphabetically, etc). User means the order the operator entered them in is preserved. "ordered-by user" is meaningful when the order among the elements have significance, e.g. DNS server search order or firewall rules.

Keys



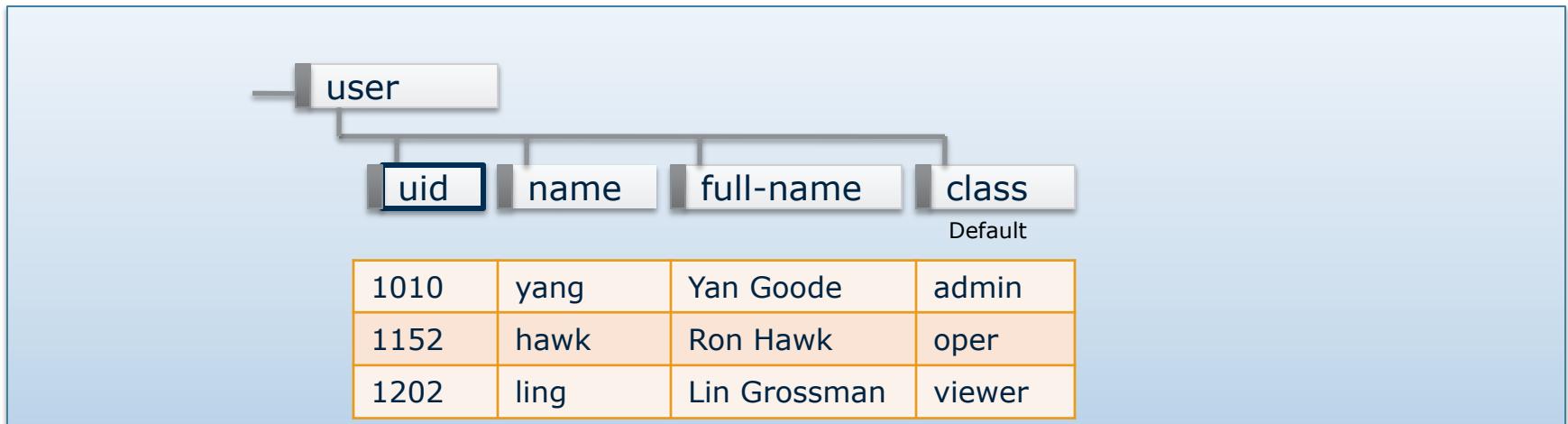
The key field is used to specify which row we're talking about.

No two rows can have same key value

```
/user[name='yang']/name = yang
/user[name='yang']/uid = 1010
/user[name='yang']/class = admin
```

```
/user[name='ling']/class = viewer
```

Keys

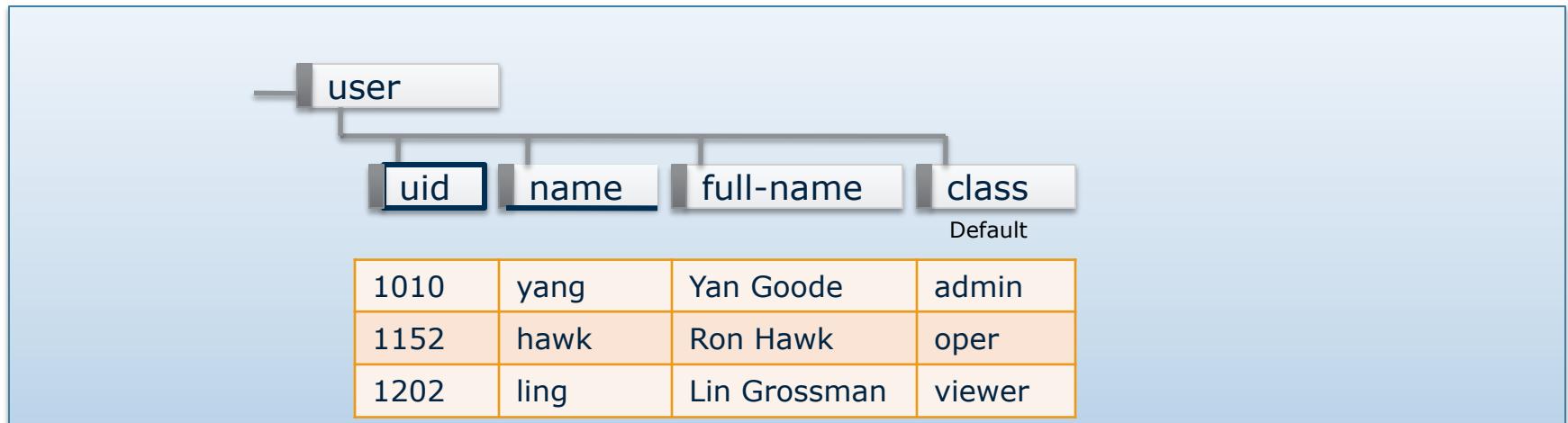


If we want, we could select the uid to be key instead.

```
/user[uid='1010']/name = yang  
/user[uid='1010']/uid = 1010  
/user[uid='1010']/class = admin
```

```
/user[uid='1202']/class = viewer
```

Unique Statement



Non- key fields can also be declared unique.

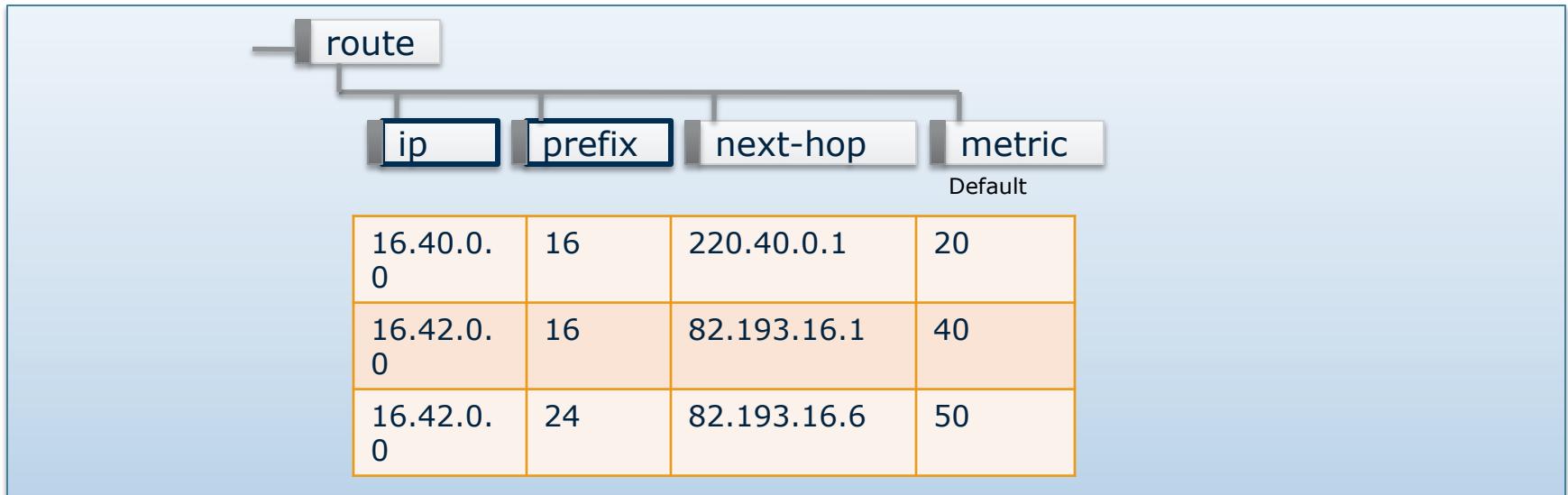
Multiple fields can be declared unique separately or in combination

```
list user {
    key uid;
    unique name;
```

...

No two rows above can have same uid, nor name

Multiple keys



Multiple key fields are needed when a single key field isn't unique.

Key fields must be a unique combination

```
list route {
    key "ip prefix";
    ...
```

```
/route[ip='16.40.0.0'][prefix='16']/next-
hop
= 220.40.0.1
```

Key order significant

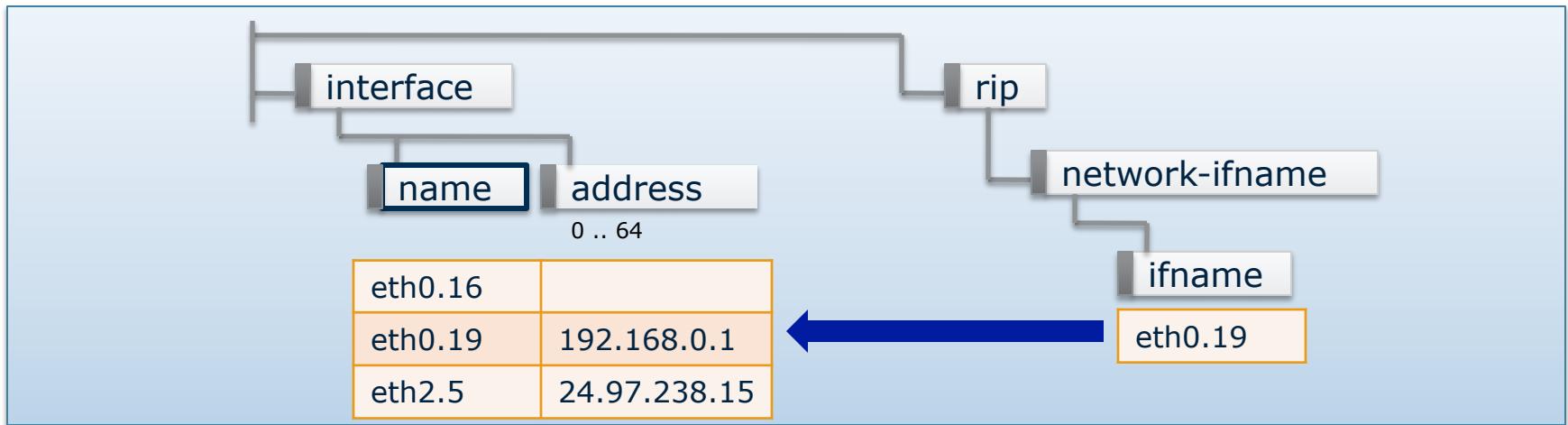
Leafref

To make an element reference one of the rows in a list, set the element type to leafref

For lists with multiple keys, the #leafrefs must match #keys in list

- A valid leafref can never be null/empty
 - But the parent leaf can be optional
- A valid leafref can never point to a row that has been deleted or renamed
- System checks validity of leafrefs automatically

Leafref



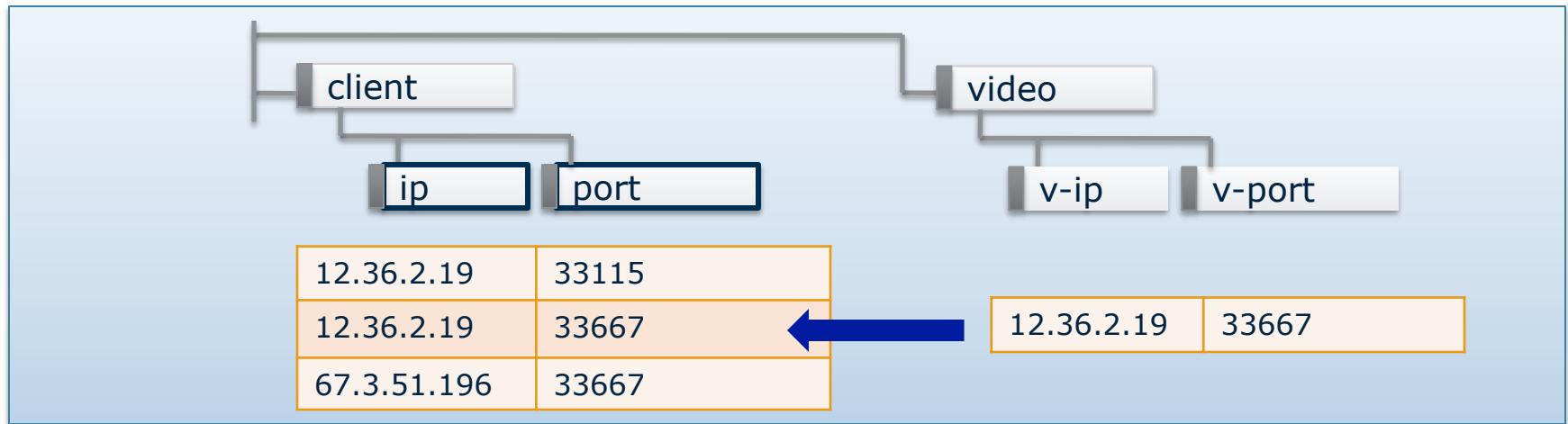
Here, the RIP routing subsystem has a list of leafrefs pointing out existing interfaces

```

container rip {
  list network-ifname {
    key ifname;

    leaf ifname {
      type leafref {
        path "/interface/name";
      }
    }
  }
}
  
```

Multiple Key Leafref



```
container video {
    leaf v-ip {
        type leafref {
            path "/client/ip";
        }
    }
    leaf v-port {
        type leafref {
            path "/client[ip=current()../v-ip]/port";
        }
    }
}
```

Deref() XPATH Operator

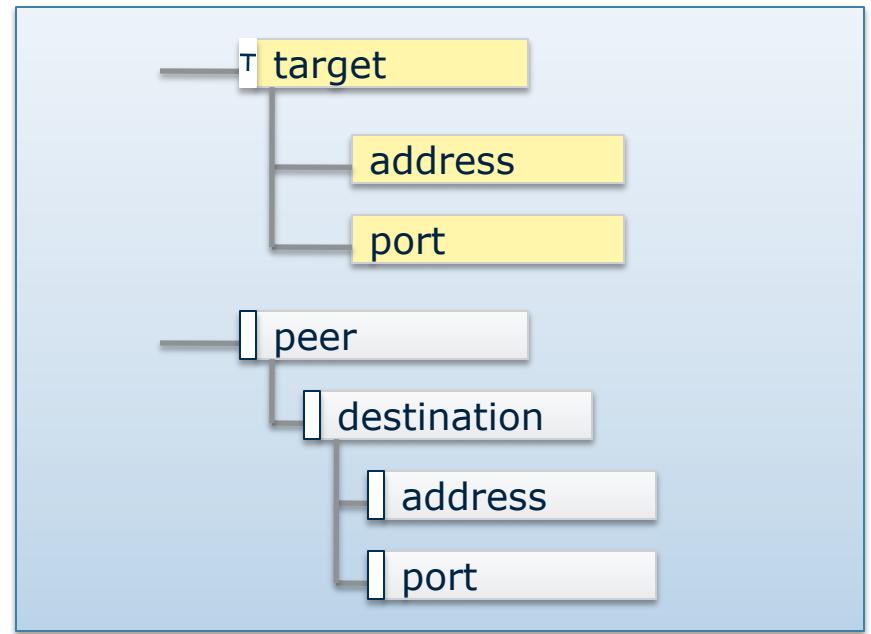
```
container video {
    leaf v-ip {
        type leafref {
            path "/client/ip";
        }
    }
    leaf v-port {
        type leafref {
            path "/client
[ip=current()/../v-ip]/port";
        }
    }
    leaf v-stream {
        type leafref {
            path "/client
[ip=current()/..v-ip]
[port=current()/..v-port]
/stream";
        }
    }
}
```

```
container video-deref {
    leaf v-ip {
        type leafref {
            path "/client/ip";
        }
    }
    leaf v-port {
        type leafref {
            path "deref(..v-ip)
/..port";
        }
    }
    leaf v-stream {
        type leafref {
            path "deref(..v-port)
/..stream";
        }
    }
}
```

Grouping Statement

Think “macro expansion”

```
grouping target {
    leaf address {
        type inet:ip-address;
        description "Target IP";
    }
    leaf port {
        type inet:port-number;
        description
            "Target port number";
    }
}
container peer {
    container destination {
        uses target;
    }
}
```



Groupings can be refined when used

Grouping Statement with Refine

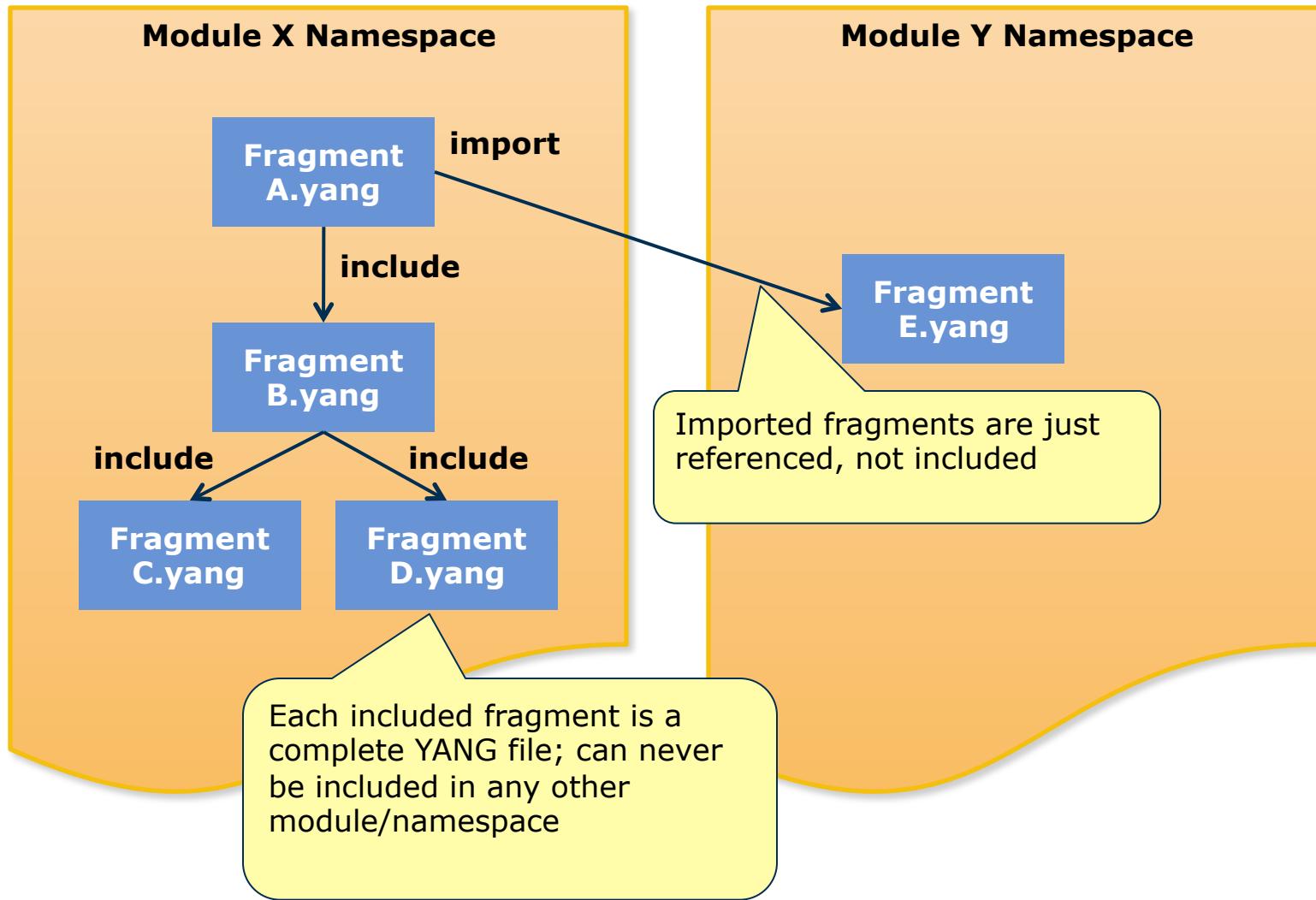
Groupings may be refined when used

```
grouping target {  
    leaf address {  
        type inet:ip-address;  
        description "Target IP";  
    }  
    leaf port {  
        type inet:port-number;  
        description  
            "Target port number";  
    }  
}
```

```
container servers {  
    container http {  
        uses target {  
            refine port {  
                default 80;  
            }  
        }  
    }  
}
```

IMPORT AND INCLUDE

Imports & Includes



Submodules

```
module acme-module {  
    namespace "...";  
    prefix acme;  
  
    import "ietf-yang-types" {  
        prefix yang;  
    }  
    include "acme-system";
```

```
submodule acme-system {  
    belongs-to acme-module {  
        prefix acme;  
    }  
  
    import "ietf-yang-types" {  
        prefix yang;  
    }  
  
    container system {  
        ...  
    }  
}
```

Each submodule belongs to one specific main module

Attention: The submodule cannot reference definitions in main module

YANG Types

YANG Base Types

- Most YANG elements have a data type
- Type may be a base type or derived type
 - Derived types may be simple typedefs or groupings (structures)
 - There are 20+ base types to start with
- And more in modules like
 - `ietf-yang-types`, RFC 6021

Type Name	Meaning
<code>int8/16/32/64</code>	Integer
<code>uint8/16/32/64</code>	Unsigned integer
<code>decimal64</code>	Non-integer
<code>string</code>	Unicode string
<code>enumeration</code>	Set of alternatives
<code>boolean</code>	True or false
<code>bits</code>	Boolean array
<code>binary</code>	Binary BLOB
<code>leafref</code>	Reference “pointer”
<code>identityref</code>	Unique identity
<code>empty</code>	No value, void
	...and more

Typedef Statement

Defines a new simple type

```
typedef percent {
    type uint16 {
        range "0 .. 100";
    }
    description "Percentage";
}

leaf completed {
    type percent;
}
```

— T percent

— | completed

Can be scoped

Type Restrictions

Integers

```
typedef my-base-int32-type {  
    type int32 {  
        range "1..4 | 10..20";  
    }  
}  
  
typedef derived-int32 {  
    type my-base-int32-type {  
        range "11..max"; // 11..20  
    }  
}
```

Strings

```
typedef my-base-str-type {  
    type string {  
        length "1..255";  
    }  
}  
  
typedef derived-str {  
    type my-base-str-type {  
        length "11 | 42..max";  
        pattern "[0-9a-fA-F]*";  
    }  
}
```

Union Statement

A value that represents one of its member types

```
typedef threshold {
    description "Threshold value in percent";
    type union {
        type uint16 {
            range "0 .. 100";
        }
        type enumeration {
            enum disabled {
                description "No threshold";
            }
        }
    }
}
```

Common YANG Types

- Commonly used YANG types defined in RFC 6021
- Use


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

 to reference these types as e.g.


```
type yang:counter64;
```

counter32/64	ipv4-address
gauge32/64	ipv6-address
object-identifier	ip-prefix
date-and-time	ipv4-prefix
timeticks	ipv6-prefix
timestamp	domain-name
phys-address	uri
ip-version	mac-address
flow-label	bridgeid
port-number	vlanid
ip-address	... and more

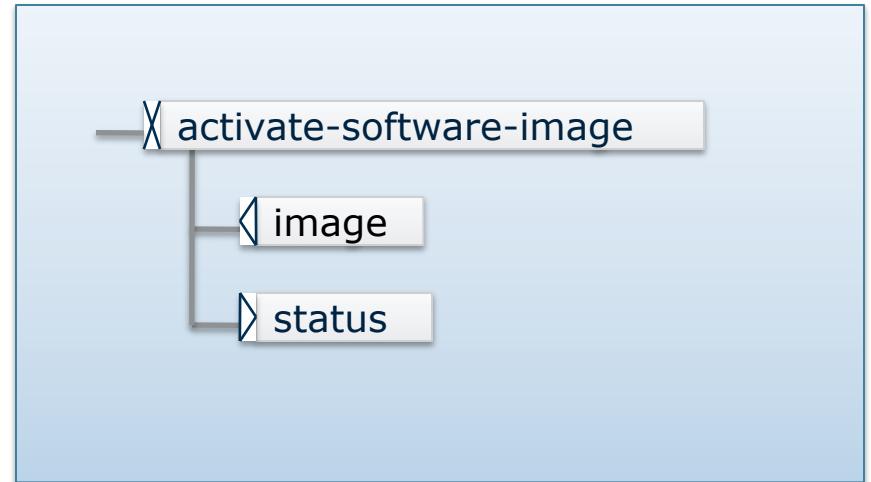
ⓘ www.rfc-editor.org/rfc/rfc6021.txt

YANG RPCs & Notifications

RPC Statement

Administrative actions with input and output parameters

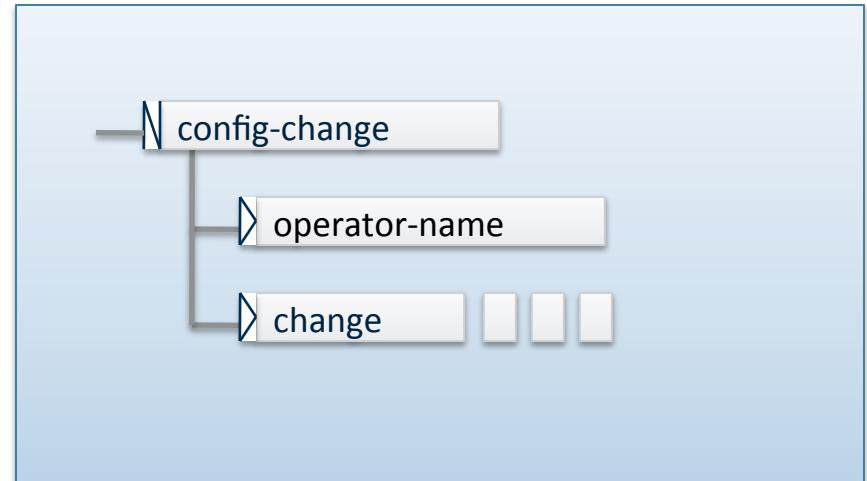
```
rpc activate-software-image {  
    input {  
        leaf image {  
            type binary;  
        }  
    }  
    output {  
        leaf status {  
            type string;  
        }  
    }  
}
```



Notification Statement

Notification with output parameters

```
notification config-change {
  description
    "The configuration changed";
  leaf operator-name {
    type string;
  }
  leaf-list change {
    type instance-identifier;
  }
}
```



Instance-identifier values

```
<change>/ex:system/ex:services/ex:ssh/ex:port</change>
<change>/ex:system/ex:user[ex:name='fred']/ex:type</change>
<change>/ex:system/ex:server[ex:ip='192.0.2.1'][ex:port='80']</change>
```

Advanced YANG Statements

Must Statement

Restricts valid values by Xpath 1.0 expression

```
container timeout {
    leaf access-timeout {
        description "Maximum time without server response";
        units seconds;
        mandatory true;
        type uint32;
    }
    leaf retry-timer {
        description "Period to retry operation";
        units seconds;
        type uint32;
        must "current() < ../access-timeout" {
            error-app-tag retry-timer-invalid;
            error-message "The retry timer must be "
                + "less than the access timeout";
        }
    }
}
```

Must Statement

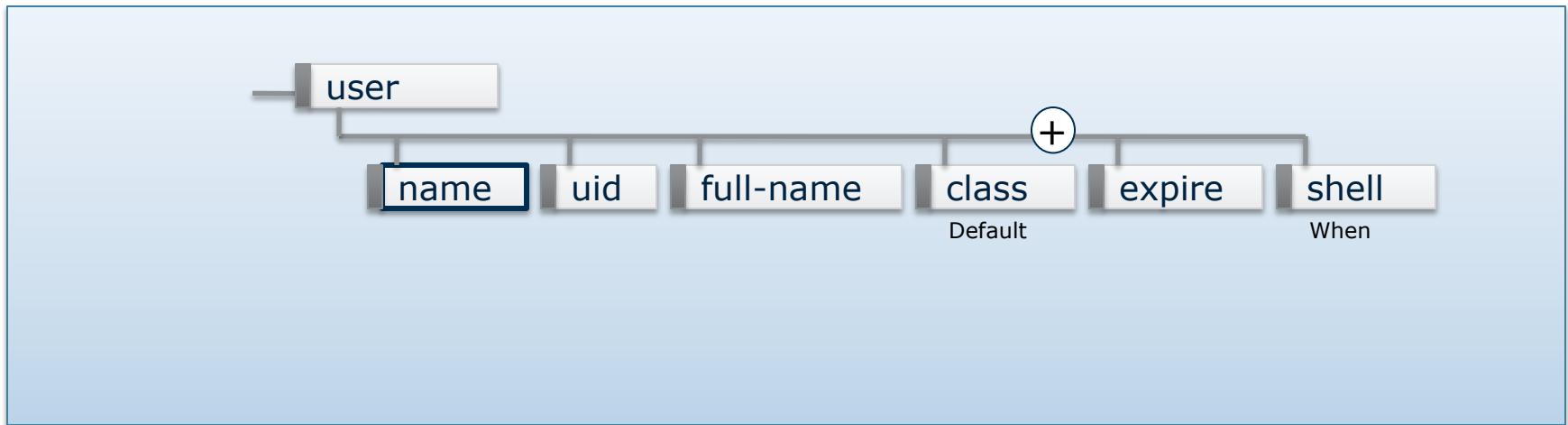
```
leaf max-weight {  
    type uint32 {  
        range "0..1000";  
    }  
    default 100;  
  
    must "sum(/sys:sys/interface[enabled = 'true']/weight)  
          < current()" {  
  
        error-message "The total weight exceeds the configured  
                      max weight";  
    }  
}
```

Augment Statement



```
augment /sys:system/sys:user {  
    leaf expire {  
        type yang:date-and-time;  
    }  
}
```

When Statement

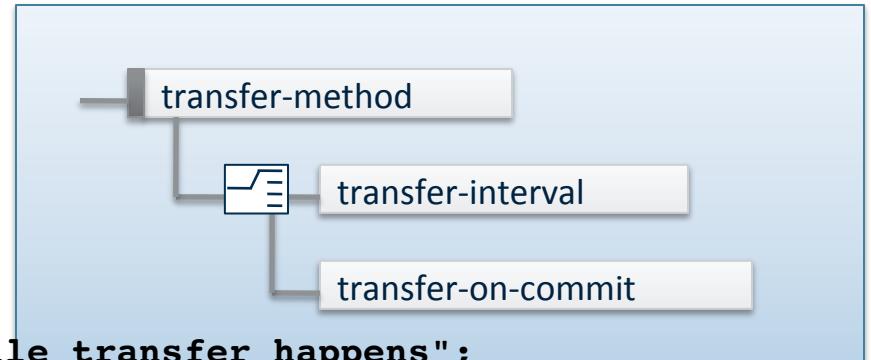


```
augment /sys:system/sys:user {
    when "sys:class = 'wheel'";
    leaf shell {
        type string;
    }
}
```

Choice Statement

Choice allows one of several alternatives

```
choice transfer-method {  
    leaf transfer-interval {  
        description "Frequency at which file transfer happens";  
        type uint16 {  
            range "15 .. 2880";  
        }  
        units minutes;  
    }  
    leaf transfer-on-commit {  
        description "Transfer after each commit";  
        type empty;  
    }  
}
```



Choice Statement

Each alternative may consist of multiple definitions

- Either as a named or anonymous group

```
choice counters {
    case four-counters {
        leaf threshold {...}
        leaf ignore-count {...}
        leaf ignore-time {...}
        leaf reset-time {...}
    }
    container warning-only {
        ...
    }
    default four-counters;
}
```

- Only in schema tree
- Not in the data tree or NETCONF
- Device handles deletion of “other” case when case is created.

Identity Statement

Identities for modeling families of related enumeration constants

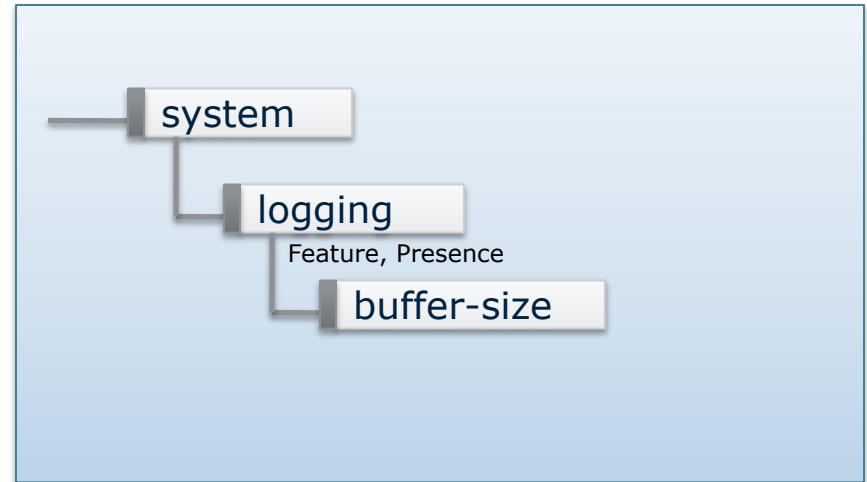
```
module phys-if {  
    ...  
    identity ethernet {  
        description  
    }  
    identity eth-1G {  
        base ethernet;  
    }  
    identity eth-10G {  
        base ethernet;  
    }  
    ...  
    module newer {  
        ...  
        identity eth-40G {  
            base phys-if:ethernet;  
        }  
        identity eth-100G {  
            base phys-if:ethernet;  
        }  
        ...  
        leaf eth-type {  
            type identityref {  
                base "phys-if:ethernet";  
            }  
        }  
    }  
}
```

Feature Statement

Mark data as conditional

```
feature has-local-disk {
    description
        "System has a local file
        system that can be used
        for storing log files";
}

container system {
    container logging {
        if-feature has-local-disk;
        presence "Logging enabled";
        leaf buffer-size {
            type filesize;
        }
    }
}
```



The features supported by a system are meant to stay relatively constant. Adding a feature is comparable to a hardware upgrade.

Deviations

Systems should conform to standard Yang Modules

- Still, if a device can't, that can be properly declared

```
deviation /base:system/base:user/base:type {
    deviate add {
        default "admin"; // new users are 'admin' by default
    }
}
deviation /base:system/base:name-server {
    deviate replace {
        max-elements 3;
    }
}
```

Thanks