

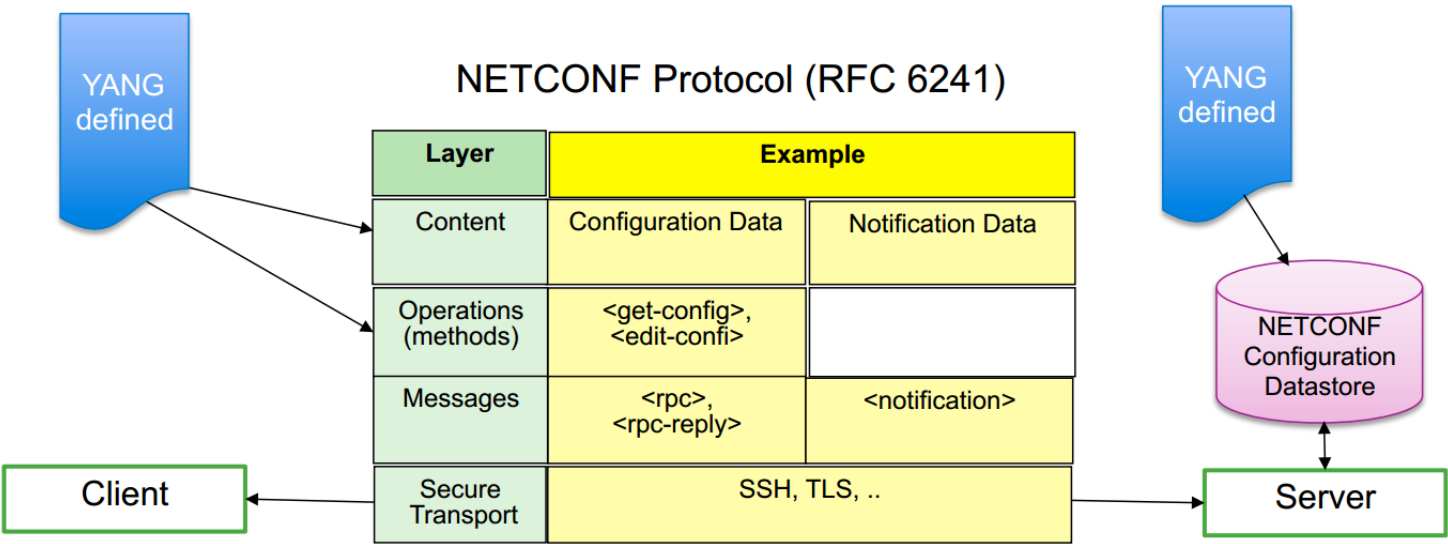
1 基础

ConfD能够读取到的YANG模型，全部是需要经过confdc编译或的fxs文件，而所有默认文件，是保存在了ConfD安装路径下的/etc/confd的路径中。而这些YANG模型的源文件会被保存在/src/confd/yang当中

YANG模型的特性：

- YANG模型和SNMP的SMI标准兼容

YANG模型的结构：



1.1 结构与基础

在编写yang的时候几点大前提注意：

- yang节点不能是数字开头
- yang的文件命名格式:

```
module-or-submodule-name ['@' revision-date] ( '.yang' / '.yin' )
```

1.1.1 module和submodule

YANG使用module进行数据建模，即一个Module就是一个完整的数据模型。一个module可以从其他module中导入数据，也可以从submodule中包含数据。

特性

- 每一个module和submodule的名字必须是唯一的
- 一个module可以被分为多个submodule（而对外来说，仍然是一个module）
- 每个submodule只能属于一个module
- module和submodule通常都以文件的形式存在，每个module或者submodule一个文件(文件名需和module名一致)

YANG编译器能够通过找到导入的module或者submodule

与Module或Submodule相关的属性：

属性名	作用
include	用于一个module包含submodule的，被包含的submodule的数据模型也会出现在数据模型中
import	用于一个module包含其他module内容的
namespace	用来唯一标识这个YANG模型与其它YANG模型不同
organization	公司名称
contact	作者名称
description	描述
feature	实现特性
revision	用来标识版本，可以有多个
identity	目的是对外宣示他的名字

重点说说include和import：

inlcude

- **include**允许module或者submodule在本module中引用其他submodule中定义的数据
- **include**不允许循环引用，比如submodule a 引用了submodule b，那么submodule b就不能再引用submodule a了

import

通过import：

- 将其他module中的grouping、typedef等引入到module中
- 将其他module中的~~！！~通过path属性

1.1.2 使用多个文件组成一个数据模型

1.2 节点类型分类

想要管理一个网络设备（基站、路由器等等），我把需要管理的内容细化成了我们平常的一些操作，那么他们所对应的YANG模型中的节点类型详见下表：

管理内容	对应于YANG模型中的节点
配置参数	container\leaf\list等等
通知上报	notification
复位设备	RPC + notification
软件升级	RPC + notification

2 配置数据节点

2.1 数据节点(data node)

有关YANG的数据节点包含的分类有：container、leaf、leaf-list、list、anyxml等等

有关数据节点，包含的类型有：

- 需要保存到数据库的节点
- 不需要保存到数据库的节点
- 一些需要保存，但是不愿意让管理站看到的节点
- 从系统中读取到的只读数据

其一：数据节点的列表一览：
(待补充)

第二：明确一下所有默认数据类型 (原文),具体会在2.2中详细说明：

Name	Description
binary	Any binary data
bits	A set of bits or flags
boolean	"true" or "false"
decimal64	64-bit signed decimal number
empty	A leaf that does not have any value
enumeration	Enumerated strings
identityref	A reference to an abstract identity
instance-identifier	References a data tree node
int8	8-bit signed integer
int16	16-bit signed integer
int32	32-bit signed integer
int64	64-bit signed integer
leafref	A reference to a leaf instance
string	Human-readable string
uint8	8-bit unsigned integer
uint16	16-bit unsigned integer
uint32	32-bit unsigned integer
uint64	64-bit unsigned integer
union	Choice of member types

2.1.1 leaf数据节点

一个leaf node包含了像integer或者string这样的简单数据。它有且仅有一个特定类型的值，并且没有子节点。

yang模型：

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

对应的报文：

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

2.1.2 container数据节点

当前理解的container是一个**不带索引的节点集合**，有点类似于C++标准库当中的set

用途1——仅用于包含节点：一个container node用来将相关的节点归总到一个subtree中，相当于最后将所有数据都放到了一个XML标签当中

```
container system {
  container login {
    leaf Username {
      type string;
      description
        "Message given at start of login name";
    }

    leaf Password {
      type string;
      description
        "Message given at start of login password";
    }
  }
}
```

对应的报文是：

```
<system>
  <login>
    <Username>Guoliang</Username>
    <Password>1234</Password>
  </login>
</system>
```

2.1.3 list数据节点

当前对list的理解是，一个**带有索引（主键）的节点集合**，与container不同，container是不带有索引的，和标准库中的map有点像

```
list user {
  key "name";

  leaf name {
    type string;
  }

  leaf full-name {
    type string;
  }

  leaf class {
    type string;
  }
}
```

对应的NetConf报文如下：

```

<user>
  <name>glocks</name>
  <full-name>Goldie Locks</full-name>
  <class>intruder</class>
</user>
<user>
  <name>snowey</name>
  <full-name>Snow White</full-name>
  <class>free-loader</class>
</user>
<user>
  <name>rzell</name>
  <full-name>Rapun Zell</full-name>
  <class>tower</class>
</user>

```

2.1.4 augment 扩展数据模型

augment主要用于拓展import其他module中容器节点的功能

以下举例几种用法：

augment..when 句式 增强list节点

增强一个list节点：

```

// 增强 "/hw:hardware/hw:component" (他会在这个路径之上增加o-ran-name)
// 如果 /hardware/component/class 的类型为 O-RAN-RADIO 以及 port 的时候才能存在节点
// must 用于数据校验
augment "/hw:hardware/hw:component" {
  when "derived-from-or-self(hw:class, 'o-ran-hw:O-RAN-RADIO') or
  derived-from-or-self(hw:class, 'ianahw:port')";
  description "New O-RAN parameters for o-ran naming";
  leaf o-ran-name {
    type leafref {
      path "/hw:hardware/hw:component/hw:name";
    }
    must "re-match(current(), '[a-zA-Z0-9][a-zA-Z0-9\\\\.\\-\\_]{0,253}[a-zA-Z0-9]')" {
      error-message "Name must match pattern and length.";
    }
    mandatory true;
    description
      "O-RAN name needed to bind and match with the name of hw element,
      to be compliant with O-RAN naming convention.";
  }
}

```

附原list节点：

```

list component {
    key name;
    description
        "List of components.
        .....
        it re-initializes itself and follow the procedure in (1).";
    reference
        "RFC 6933: Entity MIB (Version 4) - entPhysicalEntry";

    leaf name {
        type string;
        description
            "The name assigned to this component.
            This name is not required to be the same as
            entPhysicalName.";
    }

    leaf class {
        // identityref 类型详见2.2.2或上边的表格
        type identityref {
            base ianahw:hardware-class;
        }
        mandatory true;
        description
            "An indication of the general hardware type of the
            component.";
        reference
            "RFC 6933: Entity MIB (Version 4) - entPhysicalClass";
    }

    .....
}

```

独立的augment句式 增强notification

```

augment "/hw:hardware-state-oper-enabled" {
    description "new availability state";
    leaf availability-state {
        type leafref {
            path "/hw:hardware/hw:component/hw:state/o-ran-hw:availability-state";
        }
        description
            "The availability-state of the O-RU.";
    }
}

```

2.1.5 choice case 分离节点类型

使用choice case 分离不能同时出现的节点

2.1.6 identity数据类型

identity有身份、特性的意思（从目前我个人的理解，identity和枚举类型基本相似，但是应用更加灵活，能够在任何地方增加，并且附加条件判断等等）

identity ... base 句式

在一个yang文件当中声明多个identity：

```
// 一个identity的根节点
identity hardware-class {
    description
        "This identity is the base for all hardware class
        identifiers.";
}

identity unknown {
    base ianahw:hardware-class;
    description
        "This identity is applicable if the hardware class is unknown
        to the server.";
}

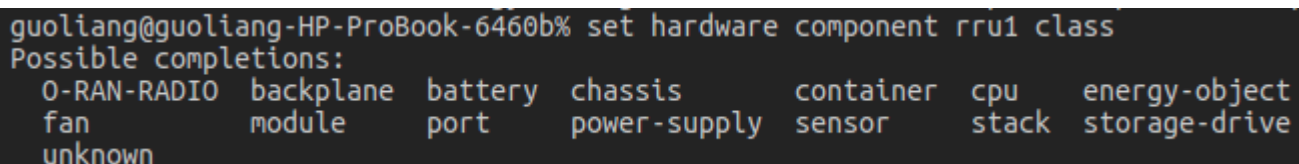
identity chassis {
    base ianahw:hardware-class;
    description
        "This identity is applicable if the hardware class is an
        overall container for networking equipment. Any class of
        physical component, except a stack, may be contained within a
        chassis; a chassis may only be contained within a stack.";
}

identity backplane {
    base ianahw:hardware-class;
    description
        "This identity is applicable if the hardware class is some sort
        of device for aggregating and forwarding networking traffic,
        such as a shared backplane in a modular ethernet switch. Note
        that an implementation may model a backplane as a single
        physical component, which is actually implemented as multiple
        discrete physical components (within a chassis or stack).";
}
```

每个identity节点最终会通过identityref数据类型进行使用

```
leaf class {
    type identityref {
        base ianahw:hardware-class;
    }
    mandatory true;
    description
        "An indication of the general hardware type of the
        component.";
    reference
        "RFC 6933: Entity MIB (Version 4) - entPhysicalClass";
}
```

效果如下:



```
guoliang@guoliang-HP-ProBook-6460b% set hardware component rru1 class
Possible completions:
  O-RAN-RADIO  backplane  battery  chassis  container  cpu  energy-object
  fan          module    port     power-supply  sensor  stack  storage-drive
  unknown
```

2.2 数据类型

2.2.1 默认支持的数据类型

这个表上节已经有了：

Name	Description
binary	Any binary data
bits	A set of bits or flags
boolean	"true" or "false"
decimal64	64-bit signed decimal number
empty	A leaf that does not have any value
enumeration	Enumerated strings
identityref	A reference to an abstract identity
instance-identifier	References a data tree node
int8	8-bit signed integer
int16	16-bit signed integer
int32	32-bit signed integer
int64	64-bit signed integer
leafref	A reference to a leaf instance
string	Human-readable string
uint8	8-bit unsigned integer
uint16	16-bit unsigned integer
uint32	32-bit unsigned integer
uint64	64-bit unsigned integer
union	Choice of member types

以下是几种不常用的类型说明：

类型名称	详细说明	应用场景
decimal64	64位带符号的十进制数，是个小数	YANG内置的唯一一个小数类型
empty	标识某个配置是否存在，而不关心其具体的值	enable节点是empty的，业务读取模型的时候，只需要知道这个状态，而不关心它的值
leafref	引用其他节点内容	存在一个服务器表、一个客户端表，那么客户端和服务之间的映射关系，即客户端只能对应已经存在的服务器表，就可以将客户端中的服务器用leafref引到服务器表的key
identityref	与identity数据节点类型配合	XXXXXX
instance-identifier	直接引用树型结构数据库中的一个叶子节点	形象的例子：歌曲库中用户建立的播放列表，列表中的每首歌都是歌曲节点的引用

leafref数据类型

identityref数据类型

instance-identifier数据类型

instance-identifier是数据库树型节点的引用，可以将任何数据库中的任意一个叶子节点引用到这个节点之上以播放列表为例子，阐述instance-identifier

yang模型:

```
list song {
    key index;
    ordered-by user;

    description
        "Example nested configuration data resource.";

    leaf index {    // not really needed
        type uint32;
        description
            "An arbitrary integer index for this playlist song.";
    }
    leaf id {
        type instance-identifier;
        mandatory true;
        description
            "Song identifier. Must identify an instance of
            /jukebox/library/artist/album/song/name.";
    }
}
```

查询实例：

```
playlist Foo-One {
    song 1 {
        id "/jukebox/library/artist[name='Foo Fighters']/album[name='Wasting Light']/song[name='A Matter Of Time']";
    }
    song 2 {
        id "/jukebox/library/artist[name='Foo Fighters']/album[name='Wasting Light']/song[name='Back And Forth']";
    }
    song 3 {
        id "/jukebox/library/artist[name='Foo Fighters']/album[name='Wasting Light']/song[name='Walk']";
    }
    song 4 {
        id "/jukebox/library/artist[name='Foo Fighters']/album[name='Wasting Light']/song[name='Arlandria']";
    }
    song 5 {
        id "/jukebox/library/artist[name='Foo Fighters']/album[name='Wasting Light']/song[name='These Days']";
    }
}
playlist Two {
    song 1 {
        id /jukebox/player;
    }
}
```

2.2.2 数据类型在不同环境下的映射与说明

YANG类型	C语言类型	SNMP (SMIV2)
int8	int8_t	Integer32
...
uint8	u_int8_t	Unsigned32
...
int64	int64_t	OCTET STRING
uint64	u_int64_t	OCTET STRING
decimal64	自定义或参考对应协议栈源码	OCTET STRING
string	自定义或参考对应协议栈源码	OCTET STRING
boolean	int	TruthValue
enumeration	int32_t	INTEGER
bits	u_int32_t, u_int64_t, 自定义	Unsigned32 or OCTET STRING
binary	自定义或参考对应协议栈源码	OCTET STRING
identityref	自定义或参考对应协议栈源码	OCTET STRING
union	XXX	OCTET STRING
instance-identifier	自定义或参考对应协议栈源码	OCTET STRING

2.3 自定义数据

2.3.1 typedef自定义数据类型 (Derived Type)

typedef和C语言当中的typedef的用法几乎一致，可以支持用户自定义数据类型

2.3.2 grouping可重用节点组 (Reusable Node Groups)

group不是一种数据类型，它的方便之处在于能够快速的被引用,group一般都用在同一个module或者换句话说，一般都用在同一个namespace当中

2.3.3 typedef和grouping的区别

group和typedef都可以通过import直接引用其他module定义好的grouping，举个比较形象的例子：

- typedef和C语言中的typedef一样，是用来定义自定义数据类型的
- grouping则有点类似于C语言中的#define，在他后边定义的内容，在调用的地方会直接被替换

2.4 节点的属性

常用默认节点属性

节点属性	说明
key	这个节点的索引（主键），用于数据节点list
config [false / true]	表示这个数据节点不是一个配置数据，但是用户可以查看
require-instance [false / true]	????
mandatory [false / true]	表示这个节点必须有值，不能和default同时存在，值为true时，强制用户必须进行配置，否则用户下发配置不生效
default	默认值

2.5 ConfD中的特殊节点属性

以下仅针对思科公司的ConfD对yang模型的扩展属性，而不是NETCONF标准，yang模型默认的属性，如果不是使用ConfD这套框架的请忽略这一小节

同样先来一表格索引，涉及复杂的单独附加说明：

节点属性	说明
tailf:callpoint [node name]	可以让外部函数
tailf:export	??

callpoint

yang数据模型可以通过带有callpoint的注释来指明回调函数。即callpoint的名称，以后可以由外部程序用来连接到该命名点

只要在yang中声明了callpoint，那么则必须在APP中注册这个callpoint的回调函数，否则会出现读取写入失败的情况

与callpoint相关的子属性：

callpoint子属性	说明
tailf:config	??
tailf:transform	??
tailf:set-hook	??
tailf:transaction-hook	??

最简单应用1：

存在一个yang模型：

```

container arpentries {
    config false;
    tailf:callpoint arpe;

    list arpe {
        key "ip ifname";
        max-elements 1024;

        leaf ip {
            type inet:ip-address;
        }

        .....
    }
}

```

由于在callpoint之下存在list，所以在代码中去实现这个callpoint如下：

```

// _____callpoint的用法
// 在程序中，设置yang模型的callpoint点的回调函数
memset(&data, 0, sizeof(struct confd_data_cbs));
data.get_elem = get_elem; // 获取值内容的回调函数
data.get_next = get_next; // 获取list索引的回调函数
strcpy(data.callpoint, arpe__callpointid_arpe);

.....

if (confd_register_data_cb(dctx, &data) == CONFD_ERR)
    confd_fatal("Failed to register data cb \n");

```

2.6 高级应用

高级应用包括一些数据校验和限等功能

2.6.1 通过自己编辑树型结构访问具体节点

2.6.1.1 直接访问节点

在yang文件中，可以通过编辑一个树型结构去访问不在同一个树枝中的某一个叶子节点，其形式如下：

- 可以通过完整路径直接访问其他模块中的节点，通过"/"符号访问下一级路径节点

```

import certus-5gnr-du-devicebaseinfo { prefix baseinfo; }

list cellULFrequencyInfo {

    tailf:snmp-oid ".6";
    tailf:snmp-name cellULFrequencyInfo;
    tailf:sort-order snmp;
    tailf:snmp-row-status-column 4;

    // todo 存在问题
    // 如果 /baseinfo:devicebaseinfo/baseinfo:dubaseinfo/baseinfo:dusupportmaxcell 这么写, 只能读取到一个数值
    must "cellULFrequencyInfo-cellid <=
/baseinfo:devicebaseinfo/baseinfo:dubaseinfo/baseinfo:dusupportmaxcell" {
        error-message "out of cell num";
    }

    key "cellULFrequencyInfo-duid cellULFrequencyInfo-cellid cellULFrequencyInfoId";

    leaf cellULFrequencyInfo-duid {
        type leafref {
            path "/baseinfo:devicebaseinfo/baseinfo:dubaseinfo/baseinfo:duid";
        }
        tailf:snmp-oid ".1";
        tailf:snmp-name cellULFrequencyInfo-duid;
    }

    .....
}

```

- 通过**相对路径访问**本模块中的其他节点

```

list neighbor {
    tailf:info "Specify a neighbor router";

    must "count(*) > 1" {
        tailf:dependency ".";
    }

    key "id";
    leaf id {
        type neighbor-id-type;
    }

    leaf remote-as {
        tailf:cli-full-command;
        tailf:info "Specify a BGP neighbor";
        type uint16 {
            range "1..65535";
            tailf:info "<1-65535> AS of remote neighbor";
        }
    }

    leaf activate {
        tailf:info "Enable the Address Family for this Neighbor";
        tailf:cli-full-command;
        must "(../remote-as or ../peer-group)";
        type empty;
    }
}

```

2.6.1.2 访问带有索引的节点

2.6.2 when的用法

2.6.3 must的用法

must语句后边可以：

- 跟随完整或相对的树形结构路径
- 路径节点后加上"[]"方括号中编写表达式（ 有点类似于C语言中的"if()" ）
- 可以使用mod \ + \ - \ < \ > 等等数学符号
- 可以使用not() sum() not()等系统定义好的函数

使用must比较全的例子在此（ 思科ConfD给出的Example ）：

- rest/basic/dhcp.yang
- validate/xpath_must/sys.yang

2.6.3.1 使用current()获取当前节点

例子：

- default-lease-time 的数值不能大于 max-lease-time

```
leaf max-lease-time {
    type uint32;
    units seconds;
    default 7200;
}

leaf default-lease-time {
    type uint32;
    units seconds;
    must 'current() <= ../max-lease-time' {
        error-message
            "The default-lease-time must be less than max-lease-time";
    }
    default 600;
}
```

2.6.3.2 使用计算符号判断

- 由此例也能看到，current()也可以通过 "." 来替代

```
must "(. mod 2) = 1" {
    error-message "can only be an odd number";
}
```

2.6.3.3 使用count统计或判断

- 判断一个leaf-list中的数值必须唯一

```

leaf-list ipv4addr {
    type inet:ipv4-address;
    max-elements 8;
    // This expression ensures that an address is unique over all interfaces
    must "count(/sys/ifc/ipv4addr[. = current()]) = 1" {
        error-message "Address has to be unique";
    }
}

```

2.6.3.4 使用sum统计或判断

- 统计ifc中有效weight个数必须要小于1000

```

must "sum(/sys/ifc[enabled = 'true']/weight) < 1000" {
    error-message "The total weight must not exceed 1000";
}

```

2.6.3.5 使用not判断

- string设置的名字不能和ifc中已经存在的名字重复
- 需要注意，虽然ifc是一个list，但是这里没有索引的概念，即string的名字不能是list ifc中任何一个的名字

```

list ifc {
    key name;
    max-elements 1024;

    leaf name {
        type interface-name;
    }

    .....

list labels {
    key string;
    max-elements 16;

    leaf string {
        type string {
            length "1..31";
            pattern "[a-zA-Z][a-zA-Z0-9_-]*";
        }
        must "not(/sys:sys/sys:ifc[name = current()])" {
            error-message
                "A label can not be the same as an interface name";
        }
    }
}
}

```

3 通知\告警 notification

3.1 在yang中定义notification

netconf协议通过定义yang中的notification来上报通知或者告警

- 以下是一个notification的yang模型

```
notification linkUp {  
  
    leaf ifIndex {  
        type leafref {  
            path "/interfaces/interface/ifIndex";  
        }  
        mandatory true;  
    }  
  
    leaf extraId {  
        type string;  
    }  
  
    list linkProperty {  
        max-elements 64;  
        leaf newlyAdded {  
            type empty;  
        }  
        leaf flags {  
            type uint32;  
            default 0;  
        }  
        list extensions {  
            max-elements 64;  
            leaf name {  
                type uint32;  
                mandatory true;  
            }  
            leaf value {  
                type uint32;  
                mandatory true;  
            }  
        }  
    }  
}
```

- 对应的notification的报文


```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2019-03-31T12:54:19.35551+00:00</eventTime>
  <linkUp xmlns="http://tail-f.com/ns/test/notif">
    <ifIndex>2</ifIndex>
    <linkProperty>
      <newlyAdded/>
      <flags>42</flags>
      <extensions>
        <name>1</name>
        <value>3</value>
      </extensions>
      <extensions>
        <name>2</name>
        <value>4668</value>
      </extensions>
    </linkProperty>
  </linkUp>
</notification>
```

4 升级、复位等远程调用

4.1 什么是RPC

附录

参考文章

- ietf:Network Modeling (netmod)
<https://datatracker.ietf.org/wg/netmod/charter/>
- itef:YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)
<https://tools.ietf.org/html/rfc6020>
- 从零开始学OpenDaylight之六：YANG
<https://www.cnblogs.com/FrankZhou2017/p/7258264.html>