I2RS RIB Route Example

Sue Hares

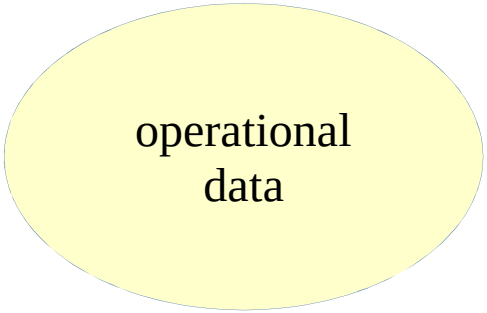
I2RS RIB Example

- Attempt to match Andy slides with I2RS RIB model (8/31/2015)
- Boiled down to IPv4 route
  - 128.2/16 with nexthop 1 – added by netconf config
  - 128.2/16 with nexthop 2 – added by I2RS RIB
  - DDOS attack causes you to overwrite NETCONF config with I2RS RIB route

Current Datastores

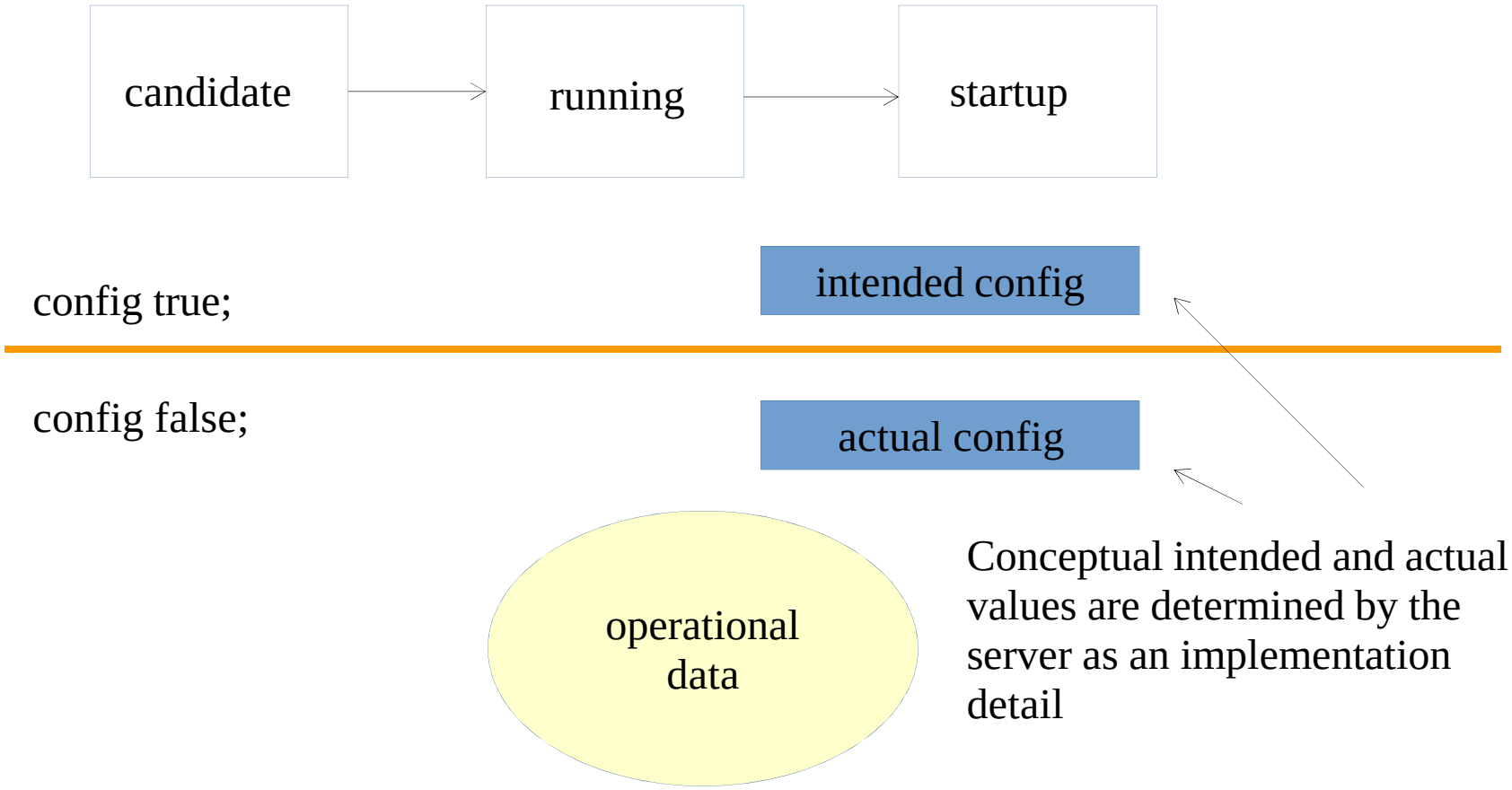
config true;

config false;



All operational data exists alongside config=true but there is no datastore defined for config=false data nodes

Current Datastores (Ext. 1)



Route

```
module i2rs-rib {
    …
    container routing-instance {
     …
      list rib-list {
       …
         list route-list {
             key “route-index”;
             uses route;
         }
    }
}
```

operational data

Extensions

```
grouping route  {
  description
    “The common attribute used for all routes;”
   uses route-prefix;
    container nexthop {
            uses nexthop;
        }
        container route-statistics {
            leaf route-state {
              type route-state-def;
              config false;    /* operational state */
             }
            leaf route-installed state {
              type route-installed-state def;
              config false;
             }
            leaf route-reason {
             type route-reason-def;
              config false;
             }
        }
     container router-attributes {
         uses router-attributes;
       }
     container route-vendor-attributes
         uses route-vendor attributes;
   }
```

Route

Index for route direct reference without prefix match; Main key.

Type: ipv4, ipv6, mpls, mac, interface

Type: v4 prefix match

Index for nexthop direct index without match

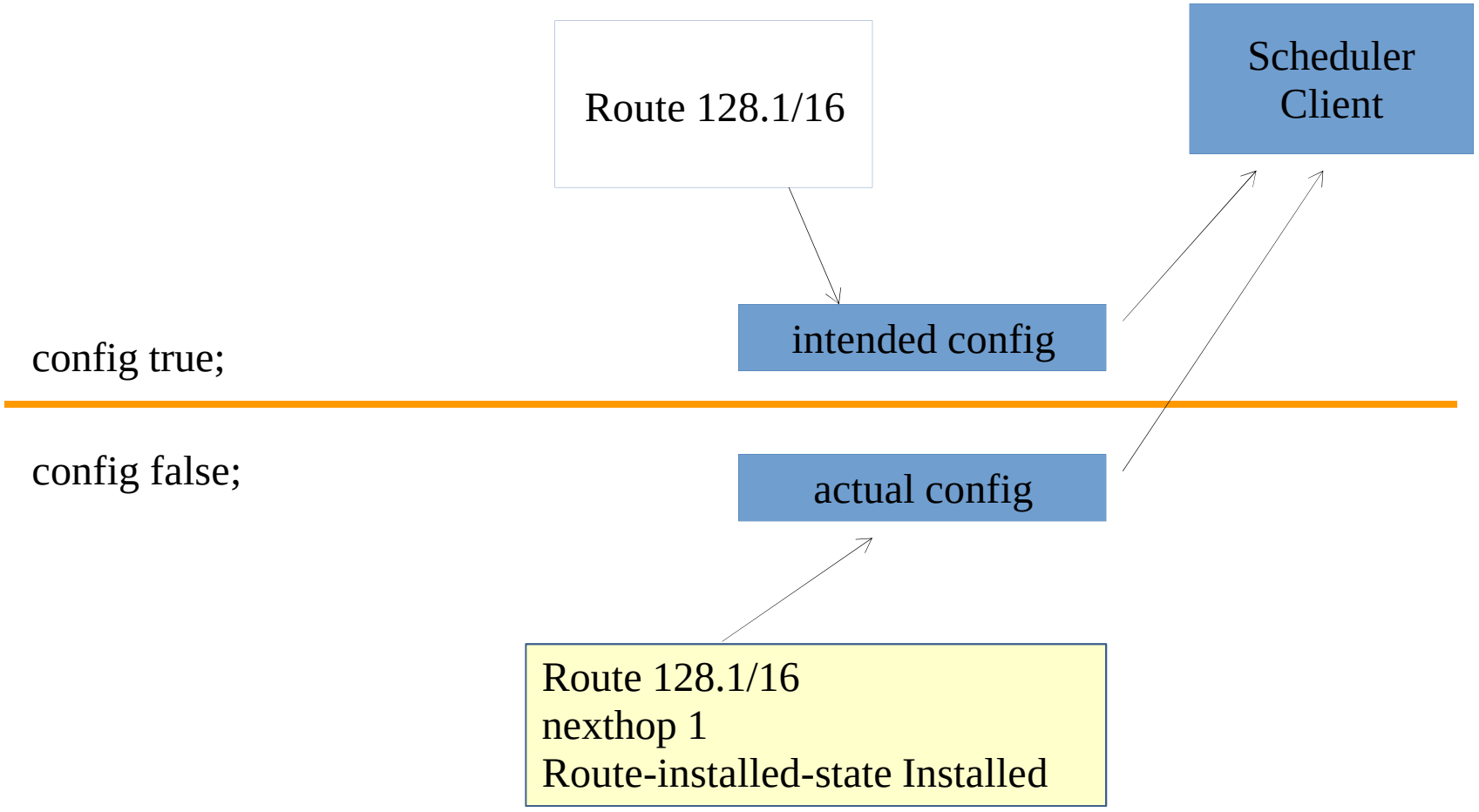
IPv4 prefix

```
module i2rs-rib { ….
 container routing-instance { …
      list rib-list { ….
         list route-list {
           key “route-index”;
           leaf route-index {
            type uint64;
            mandatory true;
           }
           leaf route-type {
            type route-type-def;
            mandatory true;
           }
          Container match {
            choice rib-route-type {….
               leaf destination-ip-v4-prefix {
                  type inet:ipv4-prefix;
                  mandatory true;
               }
            }
          }
           leaf nexthop-id {
              type uint32;
              mandatory true;
           }
           leaf next-hopo-ipv4-address {
              type inet:ipv4-prefix;
             mandatory true
           }
    }
}
```

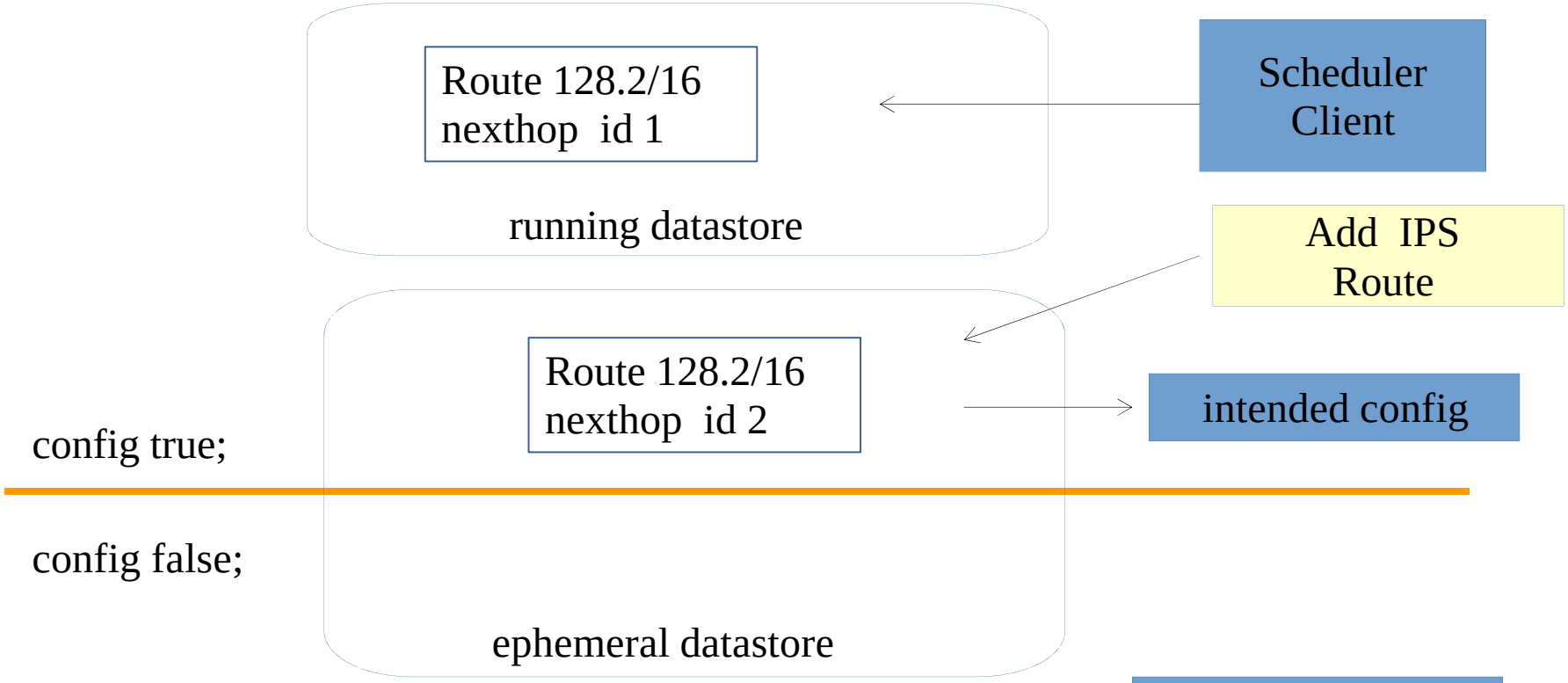
```
 container route-statistics {
leaf route-installed state {
                type route-installed-state def;
                config false;
               }
}
```

Defined as:
Installed, uninstalled

Thermostat Model Equivalent

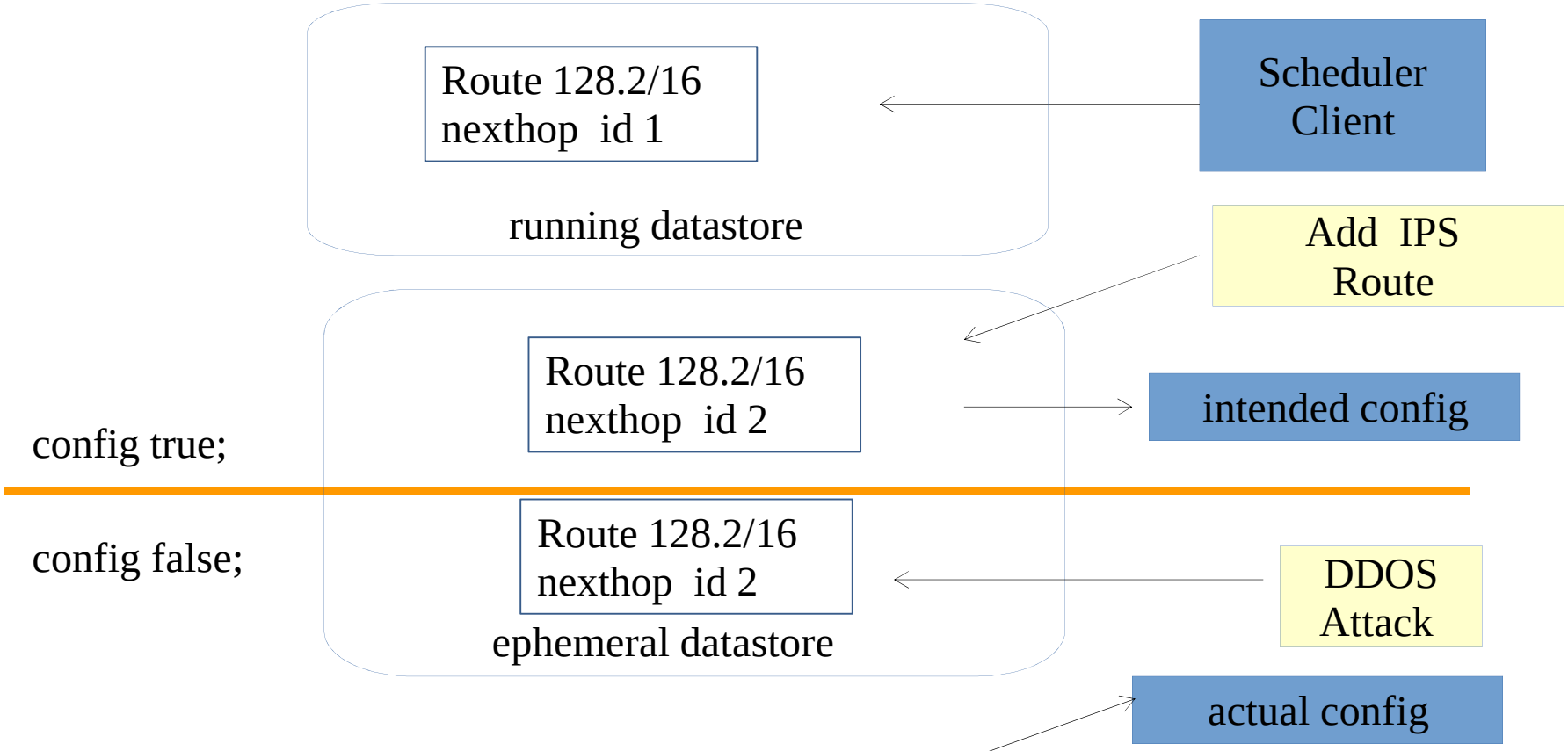


Route + Ephemeral Route



Route 128.2/16 is not deleted
by add IPS Route
the desired config is
over-ridden

Route + Ephemeral Route



Diagnostics test the DDOS response by altering the Route installed in RIB Could keep the first Route as “not installed” if implementation desired

Route 128.1/16 nexthop 2
Route-installed-state Installed
Route 128.1/16 nexthop 1
Route-installed-state Not installed

RESTCONF Example

RESTCONF Running Datastore Edit

PUT /restconf/data/i2rs-rib/instance=1/rib=IPv4/route=128.1/next-hop

{ “next-hop”:1}

RESTCONF Ephemeral Datastore Edit of config=true

PUT /restconf/data/i2rs-rib/instance=1/rib=IPv4/route=128.1/next-hop?datastore=ephemeral

{ “next-hop”:2 }

RESTCONF Ephemeral Datastore Edit of config=false

PUT /restconf/data/i2rs-rib/instance=1/rib=IPv4/route=128.1/next-hop=2/route-installed-state/datastore=ephemeral

{ “route-installed-state”: Installed }

Issues

- Did I replicate Andy's example? Did I get the RESTCONF syntax right?
- config=false should have some reflection on the true state of the route in the FIB
  - Having installed and uninstalled route makes sense for DDOS case, but it blow up the RIB size by those number
  - Implementations could immediately clean uninstalled routes out.
- Does Andy's example mean that I2RS RIB and the Routing RIB need to be aligned?
  - Right now the two are in the same general area, but not aligned.
  - Could do this if it helped make deployments easier.
- Not sure where ephemeral false would go?
- Can another client query for a list of what routes come from the I2RS RIB for the IPv4 RIB?

Summary

- Both config=true and config=false nodes can be edited in the ephemeral datastore
  - this datastore overrides normal intended config and actual config (implementation details)
- Edit and validation rules for ephemeral datastore can be different than for the running datastore
  - Actual rules TBD but cannot reference data that is “less stable” than the current context
  - Want to minimize performance overhead; maybe even provide mode where YANG validation rules are skipped

Backup on creating the shorten route

FROM I2RS YANG MODULE TO SHORT ROUTE

```
module i2rs-rib {
    …
    container routing-instance {
     …
      list rib-list {
       …
          list route-list {
              key “route-index”;
              uses route;
          }
    }
}
```

```
grouping route  {
  description
    “The common attribute
    used for all routes;”
   uses route-prefix;
    container nexthop {
            uses nexthop;
        }
     ….
 }
grouping route-prefix {
     description “common
attributes use for all routes”;
      leaf route-index {
        type uint64;
         mandatory true;
       }
       leaf route-type {
           type route-type-def;
           mandatory true;
       }
       container match {
          choice rib-route-type {
              … ipv4
              … ipv6
              … mpls
              … mac
          }
       }
```

```
grouping nexthop {
         leaf nexthop-id {
            mandatory true;
            type uint32;
         }
       choice next-hop-type {
          case next-hop base {
            list nexthop-chain {
               key “nexthop-chain-id”;
               uses nexthop-chain-member;
               }
          }
  }
```

```
case ipv4 {
   description
     “match on destination IP
      address in header”;
     container ipv4 {
         leaf ipv4-route-type {
         type ip-route-type def;
         mandatory true;
         }
       choice ip-route-type {
         case destination-ipv4-address {
           leaf destination-ipv4-prefix {
               type inet:ipv4-prefix
               mandatory true;
          }
         case destination-source-ipv4-address
           …..
     }
   }
```

Route info

```
module i2rs-rib {
    …
    container routing-instance {
     …
      list rib-list {
       …
         list route-list {
             key “route-index”;
             leaf route-index {
              type uint64;
              mandatory true;
             }
            leaf route-type {
              type route-type-def;
              mandatory true;
             }
            leaf destination-ip-v4-prefix {
               type inet:ipv4-prefix;
               mandatory true;
            }
           left nexthop-id {
               type uint32;
               mandatory true;
            }
            leaf next-hop-ipv4-address {
               type inet:ipv4-address
              mandatory true
           }
    }
}
```

IPv4 Route

```
grouping route  {
  description
    “The common attribute
    used for all routes;”
   uses route-prefix;
    container nexthop {
            uses nexthop;
        }
     ….
}
grouping route-prefix {
     description “common
 attributes use for all routes”;
      leaf route-index {
         type uint64;
          mandatory true;
       }
       leaf route-type {
            type route-type-def;
            mandatory true;
       }
       container match {
          choice route-type {
              … ipv4
              … ipv6
              … mpls
              … mac
          }
       }
```

```
grouping nexthop {
          leaf nexthop-id {
             mandatory true;
             type uint32;
          }
        choice next-hop-type {
           case next-hop base {
             list nexthop-chain {
                key “nexthop-chain-id”;
                uses nexthop-chain-member;
                }
           }
}
```

```
case ipv4 {
   description
      “match on destination IP
       address in header”;
      container ipv4 {
          leaf ipv4-route-type {
          type ip-route-type def;
          mandatory true;
          }
        choice ip-route-type {
          case destination-ipv4-address {
            leaf destination-ipv4-prefix {
                type inet:ipv4-prefix
                mandatory true;
            }
          case destination-source-ipv4-
address
              …..
      }
    }
```