I2RS RIB Route Example

Sue Hares
I2RS RIB Example

- First Case
  - 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

- candidate
- running
- startup

config true;

config false; All operational data exists alongside config=true but there is no datastore defined for config=false data nodes
module i2rs-rib {
  …
  container routing-instance {
    …
    list rib-list {
      …
      list route-list {
        key “route-index”; uses route;
      }
    }
  }
}
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

Defined as:
Installed, uninstall

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

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
}
Route initial Install

Route 128.1/16
Nexthop id 1 (192.1.1.1)

Scheduler Client

intended config

config true;

config false;

operational data

Route 128.1/16
nexthop id 1 (192.1.1.1)
Installed, active, intf 1
Route + Ephemeral Route

Route 128.2/16
nexthop id 1

running datastore

Scheduler Client

IPS Replace Route

ephemeral datastore

Route 128.2/16
nexthop id 2

intended config

config true;

config false;

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

operational data

Route 128.2/16 nexthop 2
Installed, active
Installed intf 1

Route 128.2/16 nexthop 1
Installed, non-active, intf 1
RESTCONF Example

**RESTCONF Running Datastore Edit**

PUT /restconf/data/i2rs-rib/instance=1/rib=IPv4/route=128.2/next-hop
{ "next-hop":1 }

**RESTCONF Ephemeral Datastore Edit of config=true**

PUT /restconf/data/i2rs-rib/instance=1/rib=IPv4/route=128.2/next-hop?datastore=ephemeral
{ "next-hop":2 }
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;
        }
      }
    }
  }
}

I2RS is Module based

Ephemeral true;

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

Defined as:
Installed, uninstalled

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

Route
RESTCONF Example

**RESTCONF Running Datastore Edit**

PUT /restconf/data/i2rs-rib/instance=1/rib=IPv4/route=128.2/next-hop
{ "next-hop":1 }

**RESTCONF Ephemeral Datastore Edit of config=true**

PUT /restconf/data/i2rs-rib/instance=1/rib=IPv4/route=128.2.1/next-hop?datastore=ephemeral
{ "next-hop":2 }
NETCONF Data store Get

<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get-config>
  <source>
    <emphemeral-datastore/>
  </source>
  <filter type="subtree">
    <top xmlns="http://example.com/schema/1.0/i2rs-rib/config">
      <route>
      </route>
    </top>
  </filter>
</get-config>
</rpc>
General Idea for Edit

- `<edit-config>`
  - Need priority feature indicated in merge, replace,
  - Default operations need to be: merge+priority, replace+priority
  - Error option – “all-or-nothing” (aka “rollback-on-error”)

- Still in formation
Other NETCONF changes

- No support
  - `<lock></unlock>`
  - `<confirmed commit>`

- Session I2RSClient-Agent association
  - Not related to transport
  - At association level – need graceful close and immediate close
    - “close-session” or “kill-session” which related to transport
    - I2RS need association level

- Expand commands to include ephemeral
  - get, copy-config, validate
  - Roll-back-on-error (see “all-or-nothing”)

Single Route installs

- Syntax on all routes should be checked
- Referential checks – if enabled
- Grouping – minimized
Open Issues

- Do we need Batch writes/reads beyond PUT/PATCH?
- Should we align the I2RS RIB and the Routing RIB be aligned?
  - Next hop differences prevent sharing
  - Routing RIB could have ephemeral
Backup on creating the shorten route

FROM I2RS YANG MODULE TO SHORT ROUTE
Nexthop Protection

nexthop-id - integer.

<nexthop> ::= <NEXTHOP_PROTECTION>
    
    <1> <interface-primary>
    
    <2> <interface-backup>

Protection-id 1: preference=10, nexthop-id=1
Protection-id 2: preference = 2, nexthop-id=1
Protection-id 3: preference=1, nexthop-id = 1
Protection-id 4: preference =1, nexthop-id=2
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 {
      case destination-ipv4-address {
        leaf destination-ipv4-prefix {
          type inet:ipv4-prefix
          mandatory true;
        }
      }
      case destination-source-ipv4-address
      ...
    }
  }
}


case ipv4 {
  description
  “match on destination IP  
  address in header”;
  container ipv4 {
    leaf ipv4-route-type {
      type ip-route-type def;
      mandatory true;
    }
  }
  leaf route-type {
    type route-type-def;
    mandatory true;
  }
  container match {
    choice next-hop-type {
      case next-hop base {
        list nexthop-chain {
          key “nexthop-chain-id”;  
          uses nexthop-chain-member;
        }
      }
    }
  }
}


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;
      }
    }
  }
}

18
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
                }
            }
        }
    }
    ...
}

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
        }
    }
}

case ipv4 {
    description
    “match on destination IP
    address in header”;
    container ipv4 {
        leaf ipv4-route-type {
            type ip-route-type def;
            mandatory true;
        }
        leaf route-index {
            type uint64;
            mandatory true;
        }
        container match {
            choice route-type {
                case destination-ip-v4-address {
                    leaf destination-ipv4-prefix {
                        type inet:ipv4-prefix
                        mandatory true;
                    }
                    leaf route-type {
                        type route-type-def;
                        mandatory true;
                    }
                    container match {
                        choice route-type {
                            case destination-source-ip4-
                            address
                        }
                    }
                }
            }
        }
    }
}

case next-hop {
    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;
            }
        }
    }
}...