Starting with Module 1: Routing Fundamentals.

Module 1: Routing Fundamentals

Understanding the Router's Role in Networks

Think of a router as an intelligent postal sorting facility. Just as a postal facility receives packages and decides which truck should carry them to their destination, a router receives data packets and decides which path they should take.

The Three Core Functions of a Router:

```
[PC A] -----> [Router 1] -----> [Router 2] -----> [PC B]
         ^                  ^                  ^
         |                  |                  |
     Receives           Decides Path       Forwards
     Packet            (Routing Table)      Packet
```

The Routing Table - Your Router's GPS

The routing table is like a GPS database. It contains all possible destinations and the best paths to reach them. In Junos, we call this `inet.0` for IPv4 and `inet6.0` for IPv6.

Understanding Route Types:

1. **Directly Connected Routes** (Protocol: Direct)
   - Routes to networks physically attached to the router
   - Like knowing your immediate neighbors' addresses
   - Automatically added when you configure an interface
2. **Static Routes** (Protocol: Static)
   - Routes you manually configure
   - Like memorizing specific directions to a friend's house
   - Never change unless you modify them
3. **Dynamic Routes** (Protocol: OSPF, BGP, IS-IS, etc.)
   - Routes learned from other routers
   - Like getting real-time traffic updates from other drivers
   - Can change based on network conditions

Route Preference - Breaking Ties

When multiple routes exist to the same destination, route preference (administrative distance) decides the winner. Think of it as a trust system:

```
Lower Number = More Trusted

Direct Routes:        0    (I see it myself)
Static Routes:        5    (I configured it)
OSPF Internal:       10    (My colleague told me)
RIP:                100    (Someone far away told me)
OSPF External:      150    (Heard it through the grapevine)
BGP:                170    (Internet rumor)
```

Junos CLI Pattern #1: Viewing the Routing Table

```
# Basic command structure
show route

# Why this pattern?
# - 'show' is always for viewing/verification
# - 'route' refers to the routing subsystem

# Practical variations:
show route 192.168.1.0/24    # Specific destination
show route protocol static    # Only static routes
show route table inet.0       # IPv4 routes only
show route detail            # Everything about each route
```

Visual Pattern Recognition:

```
user@router> show route

inet.0: 12 destinations, 12 routes (12 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.1.1.0/24        *[Direct/0] 00:10:29
                    > via ge-0/0/0.0    <-- Interface
172.16.1.0/24      *[Static/5] 00:05:15
                      to 10.1.1.254 via ge-0/0/0.0
192.168.1.0/24     *[OSPF/10] 00:03:22, metric 2
                    > to 10.1.1.2 via ge-0/0/0.0
```

Pattern Analysis:

- `[Protocol/Preference]` - Always in square brackets
- Time shown as `HH:MM:SS` - How long route has been active
- `>` symbol - Indicates the active next-hop
- `*` symbol - Currently active route

Longest Prefix Match - The GPS Precision Rule

Routers always choose the most specific route. This is like GPS choosing "123 Main Street" over just "Main Street" when both are available.

```
Route Table Example:
10.0.0.0/8      -> Interface A    (Covers 10.0.0.0 - 10.255.255.255)
10.1.0.0/16     -> Interface B    (Covers 10.1.0.0 - 10.1.255.255)
10.1.1.0/24     -> Interface C    (Covers 10.1.1.0 - 10.1.1.255)

Destination: 10.1.1.100
Winner: 10.1.1.0/24 (Most specific match)
```

Troubleshooting Pattern #1: Route Not Working?

The Systematic Approach:

1. **Check if route exists:**

   ```
   show route 192.168.1.0/24
   # No output? Route doesn't exist
   ```

2. **Check if route is active:**

   ```
   show route 192.168.1.0/24 detail
   # Look for "State: <Active>"
   ```

3. **Check preference conflicts:**

   ```
   show route 192.168.1.0/24 all
   # Shows hidden/inactive routes too
   ```

4. **Verify next-hop reachability:**

   ```
   show route <next-hop-ip>
   # Next-hop must be reachable
   ```

Module 2 Preview: Protocol Independent Routing

Now that you understand how routes work, we'll learn to create them manually. This is where you gain control over your router's decisions.

Key Mindset Shift:

- Module 1: Understanding how the router thinks
- Module 2: Teaching the router what you want it to think

Module 2: Protocol Independent Routing

Understanding Protocol Independent Routing

Protocol Independent Routing (PIR) refers to routing mechanisms that work regardless of which dynamic routing protocols you're running. Think of these as the "manual overrides" in your routing system.

Static Routes - The Foundation

Static routes are like permanent road signs you install. They never change unless you manually modify them.

When to Use Static Routes:

- Default routes to ISPs
- Routes to stub networks (networks with only one way in/out)
- Backup routes for critical destinations
- Lab environments and small networks

Junos CLI Pattern #2: Static Route Configuration

```
# Basic static route pattern:
set routing-options static route <destination> next-hop <gateway>

# The thought process:
# 1. "routing-options" - I'm configuring routing behavior
# 2. "static" - I'm working with static routes
# 3. "route" - I'm defining a specific route
# 4. <destination> - Where traffic should go
# 5. "next-hop" - How to get there
```

Visual Configuration Hierarchy:

```
routing-options {
    static {
        route 192.168.1.0/24 {
            next-hop 10.1.1.254;
        }
    }
}
```

Static Route Variations - Know Your Options

1. Next-hop IP Address (Most Common)

```
set routing-options static route 192.168.1.0/24 next-hop 10.1.1.254

# Traffic flow:
[Your Router] ---> [10.1.1.254] ---> [192.168.1.0/24]
```

2. Qualified Next-hop (Multiple Paths)

```
set routing-options static route 192.168.1.0/24 qualified-next-hop 10.1.1.1 preference 10
set routing-options static route 192.168.1.0/24 qualified-next-hop 10.1.1.2 preference 20

# Creates primary and backup paths:
```

```
Primary:   [Your Router] ---> [10.1.1.1] ---> [Destination]
Backup:    [Your Router] ---> [10.1.1.2] ---> [Destination]
```

3. Discard Route (Black Hole)

```
set routing-options static route 192.168.100.0/24 discard

# Use case: Preventing routing loops
# Traffic to 192.168.100.0/24 gets dropped
```

4. Reject Route (With ICMP Notification)

```
set routing-options static route 192.168.200.0/24 reject

# Similar to discard but sends ICMP unreachable
# More polite - tells sender "destination unreachable"
```

Aggregate Routes - The Summarization Tool

Aggregate routes combine multiple specific routes into one summary route. Like saying "all houses on Main Street" instead of listing each house number.

The Problem Aggregate Routes Solve:

```
Without Aggregation:
10.1.1.0/24 -> Next-hop A
10.1.2.0/24 -> Next-hop A
10.1.3.0/24 -> Next-hop A
10.1.4.0/24 -> Next-hop A

With Aggregation:
10.1.0.0/22 -> Next-hop A (Covers all four networks)
```

Junos CLI Pattern #3: Aggregate Route Configuration

```
# Basic aggregate route:
set routing-options aggregate route 10.1.0.0/22

# The pattern breakdown:
# - "aggregate" instead of "static"
# - Creates route ONLY if more specific routes exist
# - Automatically activated/deactivated
```

Advanced Aggregate Options:

```
# With specific contributing routes
set routing-options aggregate route 10.1.0.0/22 policy AGGREGATE-POLICY

# Policy to control which routes contribute:
set policy-options policy-statement AGGREGATE-POLICY term 1 from protocol direct
set policy-options policy-statement AGGREGATE-POLICY term 1 from route-filter 10.1.0.0/24 orlonger
set policy-options policy-statement AGGREGATE-POLICY term 1 then accept
```

Generated Routes - The Intelligent Summaries

Generated routes are like aggregate routes with more intelligence. They can modify attributes and exist even without contributing routes.

Key Differences:

```
Aggregate Route:
- Exists ONLY if contributing routes exist
- Inherits next-hop from contributing routes
- Simple summarization
```

```
Generated Route:
- Can exist independently
- Can set custom next-hop
- Can modify route attributes
- More flexible
```

Junos CLI Pattern #4: Generated Route Configuration

```
# Basic generated route:
set routing-options generate route 172.16.0.0/12

# With specific next-hop:
set routing-options generate route 172.16.0.0/12 next-hop 10.1.1.1

# With policy control:
set routing-options generate route 172.16.0.0/12 policy GENERATE-POLICY
```

Martian Routes - The Protection Mechanism

Martian routes are invalid routes that Junos automatically ignores. Like a bouncer at a club checking IDs.

Default Martian Addresses:

```
0.0.0.0/0 exact        # Only 0.0.0.0/32 (not default route)
127.0.0.0/8 orlonger   # Loopback addresses
128.0.0.0/16 orlonger  # Class B private (historical)
191.255.0.0/16 orlonger # Reserved
192.0.0.0/24 orlonger  # Reserved
223.255.255.0/24 orlonger # Reserved
240.0.0.0/4 orlonger   # Class E
```

Managing Martians:

```
# View current martians:
show route martians

# Add custom martian:
set routing-options martians 10.99.99.0/24 orlonger

# Remove from martian list (allow the route):
delete routing-options martians 10.0.0.0/8
```

Troubleshooting Pattern #2: Static Route Issues

The Diagnostic Flow:

1. **Route Not Appearing?**

   ```
   # Check configuration:
   show configuration routing-options static

   # Check if committed:
   show | compare
   commit check
   ```

2. **Route Inactive?**

   ```
   # Check next-hop reachability:
   show route <next-hop-ip>

   # If next-hop unreachable:
   set routing-options static route <destination> next-hop <gateway> resolve
   ```

3. **Wrong Route Selected?**

```
# Check all routes:
show route <destination> all

# Adjust preference:
set routing-options static route <destination> next-hop <gateway> preference 200
```

Lab Mindset Development

When approaching static routing configuration:

1. **Plan Before Configuring:**
   - What networks need static routes?
   - What's the next-hop for each?
   - Do I need redundancy?
2. **Verify Each Step:**

   ```
   # After each route:
   show route protocol static
   ping <destination> rapid count 5
   ```

3. **Think in Hierarchies:**
   - Use aggregate routes for scalability
   - Use generated routes for flexibility
   - Use static routes for specific needs

Common Pitfall Awareness

Pitfall 1: Next-hop Unreachable

```
# Bad:
set routing-options static route 192.168.1.0/24 next-hop 172.16.1.1
# (If 172.16.1.1 not directly connected)

# Good:
set routing-options static route 192.168.1.0/24 next-hop 172.16.1.1 resolve
```

Pitfall 2: Overlapping Routes Without Preference

```
# Problem:
set routing-options static route 10.0.0.0/8 next-hop 1.1.1.1
set routing-options static route 10.0.0.0/8 next-hop 2.2.2.2
# Both have same preference!

# Solution:
set routing-options static route 10.0.0.0/8 qualified-next-hop 1.1.1.1 preference 10
set routing-options static route 10.0.0.0/8 qualified-next-hop 2.2.2.2 preference 20
```

Module 3 Preview: OSPF Fundamentals

Now that you can manually control routing, we'll learn how routers can automatically share routing information using OSPF. This is where your network becomes self-learning.

Module 3: Fundamentals of OSPF

What is OSPF? - The Neighborhood Watch Protocol

OSPF (Open Shortest Path First) is like creating a neighborhood watch program for routers. Instead of you manually telling each router about every network (static routes), routers talk to each other and share information automatically.

The Real-World Analogy:

```
Static Routing = You calling each neighbor to tell them about a road closure
OSPF = Neighbors automatically sharing traffic updates with each other
```

How OSPF Thinks - The Link-State Concept

OSPF is a "link-state" protocol. Each router builds a complete map of the network, like having a GPS with real-time traffic data.

The Three-Step Process:

```
Step 1: Discover Neighbors
[Router A] <--Hello--> [Router B]
"Hi, I'm Router A!"    "Hi, I'm Router B!"

Step 2: Share Link Information
[Router A] --LSA--> [Router B]
"I have these networks attached"

Step 3: Calculate Best Paths
Each router independently calculates shortest paths
(Using Dijkstra's algorithm - like GPS finding shortest route)
```

OSPF Packets - The Language Routers Speak

The Five OSPF Message Types:

1. **Hello Packets** - The Heartbeat
   - Sent every 10 seconds (default)
   - "I'm alive and these are my parameters"
   - Used to discover and maintain neighbors
2. **Database Description (DBD)** - The Table of Contents
   - "Here's a summary of what I know"
   - Used during initial synchronization
3. **Link State Request (LSR)** - The Question
   - "Please send me details about these networks"
   - Used to request specific information
4. **Link State Update (LSU)** - The Answer
   - "Here's the detailed information you requested"
   - Contains actual route information
5. **Link State Acknowledgment (LSAck)** - The Receipt
   - "I received your update"
   - Ensures reliable delivery

OSPF Neighbor Formation - The Handshake Process

Visual State Machine:

```
Down -> Init -> 2-Way -> ExStart -> Exchange -> Loading -> Full

Down:     No hellos received
Init:     Hello received, but not two-way
2-Way:    Both routers see each other
ExStart:  Decide who leads the conversation
Exchange: Share database summaries
Loading:  Request missing information
Full:     Fully synchronized
```

The Designated Router (DR) - The Meeting Coordinator

On multi-access networks (like Ethernet), OSPF elects a Designated Router to reduce chatter.

Without DR (Mesh of Conversations):

```
[R1]---[R2]
 | \  / |
 |  \/  |
 |  /\  |
 | /  \ |
[R3]---[R4]
6 separate conversations needed!
```

With DR (Hub and Spoke):

```
    [R2]
     |
[R1]-[DR]-[R3]
     |
    [R4]
Only 4 conversations needed!
```

DR Election Process:

1. Highest OSPF Priority wins (default: 128)
2. If tied, highest Router ID wins
3. BDR (Backup DR) is second place

OSPF Areas - The Scalability Solution

OSPF uses areas to create hierarchical design, like organizing a large company into departments.

```
        Area 0 (Backbone)
       [ABR]---[ABR]
        /         \
    Area 1      Area 2
   [Routers]   [Routers]
```

Area Rules:

- Area 0 = Backbone (must exist)
- All areas must connect to Area 0
- ABR (Area Border Router) connects areas
- Each area maintains separate database

Junos CLI Pattern #5: Basic OSPF Configuration

```
# Step 1: Define Router ID (Optional but recommended)
set routing-options router-id 1.1.1.1

# Step 2: Configure OSPF
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0

# The pattern breakdown:
# - "protocols" = Configuring dynamic protocols
# - "ospf" = Working with OSPF
# - "area X.X.X.X" = Defining area (always in dotted decimal)
# - "interface" = Assigning interface to area
```

Complete Basic Configuration:

```
# Router ID
set routing-options router-id 10.0.0.1

# OSPF Configuration
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
```

Understanding OSPF Interface Types

1. Point-to-Point (No DR/BDR)

```
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 interface-type p2p
# Use on: Serial links, point-to-point Ethernet
```

2. Broadcast (Default for Ethernet)

```
# No configuration needed - default for Ethernet
# Elects DR/BDR
```

3. Passive Interface

```
set protocols ospf area 0.0.0.0 interface lo0.0 passive
# Advertises network but doesn't form neighbors
# Use on: Loopbacks, user-facing interfaces
```

Junos CLI Pattern #6: OSPF Verification Commands

```
# The verification toolkit:
show ospf neighbor              # Who are my neighbors?
show ospf interface             # Which interfaces run OSPF?
show ospf database              # What's in the OSPF database?
show ospf route                 # What routes did OSPF calculate?
show route protocol ospf        # What routes are in routing table?
```

Reading OSPF Neighbor Output:

```
Address         Interface       State    ID              Pri Dead
10.1.1.2        ge-0/0/0.0      Full     2.2.2.2         128  36
                                 ^^^^
                                 This is what we want!
```

State Meanings:

- Full = Perfect! Fully synchronized
- 2-Way = Normal for non-DR/BDR neighbors
- Init/ExStart/Exchange/Loading = Transitioning
- Down = Problem!

OSPF Timers - The Heartbeat Settings

Default Timers:

```
Hello Interval: 10 seconds
Dead Interval: 40 seconds (4x Hello)

If no Hello received for 40 seconds = Neighbor is declared dead
```

Modifying Timers (Must Match on Both Ends!):

```
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 hello-interval 1
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 dead-interval 4
```

OSPF Authentication - The Security Layer

```
# Simple Password
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 authentication simple-password MySecret

# MD5 (More Secure)
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 authentication md5 1 key SecretKey
```

Troubleshooting Pattern #3: OSPF Issues

The OSPF Troubleshooting Flowchart:

1. **No Neighbors?**

```
# Check 1: Is OSPF enabled on interface?
show ospf interface

# Check 2: Are Hellos being sent/received?
monitor traffic interface ge-0/0/0.0 matching "proto 89"

# Check 3: Do parameters match?
show ospf interface detail
# Look for: Area, Hello interval, Network type
```

2. **Stuck in Init/2-Way?**

```
# Check MTU mismatch:
show interface ge-0/0/0 | match MTU

# Check authentication:
show configuration protocols ospf area 0 interface ge-0/0/0.0
```

3. **Routes Not Appearing?**

```
# Check OSPF database:
show ospf database

# Check OSPF's calculated routes:
show ospf route

# Check if they made it to routing table:
show route protocol ospf
```

OSPF Metrics - Understanding Path Selection

OSPF uses "cost" as its metric. Lower cost = better path.

Default Cost Calculation:

```
Cost = Reference Bandwidth / Interface Bandwidth
Default Reference = 100 Mbps

Examples:
100 Mbps interface: Cost = 100/100 = 1
1 Gbps interface:   Cost = 100/1000 = 1 (rounds up)
10 Mbps interface:  Cost = 100/10 = 10
```

Adjusting Costs:

```
# Manual cost per interface:
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 metric 50

# Change reference bandwidth:
set protocols ospf reference-bandwidth 10g
```

Common OSPF Scenarios and Solutions

Scenario 1: Preventing Transit Traffic

```
# Make area a stub area:
set protocols ospf area 0.0.0.1 stub
# Must configure on ALL routers in the area
```

Scenario 2: Load Balancing

```
# OSPF automatically load balances equal-cost paths
# Ensure costs are equal:
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 metric 10
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0 metric 10
```

Scenario 3: Faster Convergence

```
# Enable BFD (Bidirectional Forwarding Detection):
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 bfd-liveness-detection minimum-interval 300
```

OSPF Design Best Practices

1. **Always Set Router ID**
   - Prevents issues during reboots
   - Makes troubleshooting easier
2. **Use Areas for Scale**
   - Keep areas under 50 routers
   - Summarize at area boundaries
3. **Passive Interfaces by Default**

   ```
   set protocols ospf area 0.0.0.0 interface all
   set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 disable false
   ```

4. **Document Your Design**
   - Which interfaces in which areas
   - DR/BDR placement strategy
   - Cost manipulation reasoning

Developing OSPF Instincts

When You See This:

- Neighbors stuck in Init → Check MTU
- Neighbors stuck in 2-Way → Normal if neither is DR/BDR
- Neighbors stuck in ExStart → MTU or Router ID conflict
- Neighbors flapping → Check physical layer or authentication
- Routes missing → Check area design and stub settings

Module 4 Preview: Routing Policy

Now that OSPF can share routes automatically, we need to control what gets shared and how. Routing policies are like customs officers, inspecting and modifying routes as they move between protocols.

Module 4: Routing Policy

What is Routing Policy? - The Traffic Controller

Routing policies are like customs officers at borders. They inspect routes as they move between protocols and can:

- Allow or deny routes
- Modify route attributes
- Add tags or communities
- Change preferences

The Key Concept:

```
[OSPF Domain] --Export Policy--> [Routing Table] --Import Policy--> [BGP Domain]
              "What leaves"                        "What enters"
```

Import vs Export - The Direction Matters

This is where most people get confused. Think of it from the protocol's perspective:

Export Policy:

- FROM routing table TO protocol
- "What I advertise to my neighbors"
- Controls what goes OUT of the protocol

Import Policy:

- FROM protocol TO routing table
- "What I accept from my neighbors"
- Controls what comes INTO the routing table

```
Neighbor ----> [Import Policy] ----> Protocol ----> Routing Table
                                         |
                                         v
Routing Table ----> [Export Policy] ----> Protocol ----> Neighbor
```

The Anatomy of a Routing Policy

A routing policy is built like a legal document with terms (rules) that are evaluated in order.

```
Policy-Statement: POLICY-NAME
    |
    +-- Term 1: "If traffic is from subnet X, then accept"
    |
    +-- Term 2: "If protocol is static, then reject"
    |
    +-- Term 3: "If metric > 100, then modify metric to 50"
    |
    +-- Default: "Reject everything else"
```

Junos CLI Pattern #7: Basic Policy Structure

```
# The fundamental pattern:
set policy-options policy-statement <POLICY-NAME> term <TERM-NAME> from <MATCH-CONDITION>
set policy-options policy-statement <POLICY-NAME> term <TERM-NAME> then <ACTION>

# Breaking it down:
# - policy-options: The policy configuration section
# - policy-statement: Defining a new policy
# - term: A single rule within the policy
# - from: Match conditions (if...)
# - then: Actions to take (then...)
```

Real Example:

```
# Policy to accept only 10.0.0.0/8 networks
set policy-options policy-statement ACCEPT-10-NETWORKS term 1 from route-filter 10.0.0.0/8 orlonger
set policy-options policy-statement ACCEPT-10-NETWORKS term 1 then accept
set policy-options policy-statement ACCEPT-10-NETWORKS term 2 then reject
```

Match Conditions - The "FROM" Clause

Think of these as your inspection criteria:

1. Route-Based Matching:

```
# Exact match
from route-filter 192.168.1.0/24 exact

# Prefix and longer
from route-filter 10.0.0.0/8 orlonger

# Prefix and shorter
```

```
from route-filter 10.0.0.0/8 upto /24

# Range of prefix lengths
from route-filter 10.0.0.0/8 prefix-length-range /16-/24
```

2. Protocol-Based Matching:

```
from protocol static
from protocol ospf
from protocol bgp
from protocol direct
```

3. Attribute-Based Matching:

```
from metric 100
from preference 170
from neighbor 192.168.1.1
from area 0.0.0.0
from tag 100
```

4. Multiple Conditions (Logical AND):

```
# Must match ALL conditions
from protocol ospf
from route-filter 10.0.0.0/8 orlonger
from metric 10
```

Actions - The "THEN" Clause

1. Basic Actions:

```
then accept              # Allow the route
then reject             # Deny the route silently
```

2. Modification Actions:

```
then {
    metric 100;         # Set metric
    preference 150;     # Set preference
    tag 200;           # Add tag
    community add NO-EXPORT;  # Add BGP community
    next-hop 10.1.1.1; # Change next-hop
}
```

3. Control Flow Actions:

```
then next term          # Continue to next term
then next policy        # Continue to next policy
```

Policy Chains - Multiple Policies

Policies can be chained together like filters:

```
# Apply multiple policies (evaluated in order)
set protocols ospf export [ POLICY1 POLICY2 POLICY3 ]

# Flow:
Route -> POLICY1 -> POLICY2 -> POLICY3 -> Final Decision
```

Junos CLI Pattern #8: Applying Policies

For OSPF:

```
# Export static routes into OSPF
set protocols ospf export STATIC-TO-OSPF

# Import policy (less common for OSPF)
set protocols ospf import FILTER-OSPF-ROUTES
```

For BGP:

```
# Per-neighbor policies
set protocols bgp group EXTERNAL neighbor 192.168.1.1 import CUSTOMER-IN
set protocols bgp group EXTERNAL neighbor 192.168.1.1 export CUSTOMER-OUT
```

For RIP:

```
set protocols rip group RIPGROUP export CONNECTED-TO-RIP
set protocols rip group RIPGROUP import FILTER-RIP
```

Common Policy Patterns

Pattern 1: Redistribute Connected and Static into OSPF

```
# Define the policy
set policy-options policy-statement REDIS-TO-OSPF term CONNECTED from protocol direct
set policy-options policy-statement REDIS-TO-OSPF term CONNECTED then accept
set policy-options policy-statement REDIS-TO-OSPF term STATIC from protocol static
set policy-options policy-statement REDIS-TO-OSPF term STATIC then accept
set policy-options policy-statement REDIS-TO-OSPF term DENY then reject

# Apply it
set protocols ospf export REDIS-TO-OSPF
```

Pattern 2: Filter Specific Networks

```
# Accept only specific prefixes
set policy-options policy-statement SPECIFIC-ONLY term ALLOWED from route-filter 192.168.0.0/16 orlonger
set policy-options policy-statement SPECIFIC-ONLY term ALLOWED from route-filter 172.16.0.0/12 orlonger
set policy-options policy-statement SPECIFIC-ONLY term ALLOWED then accept
set policy-options policy-statement SPECIFIC-ONLY term DENY-REST then reject
```

Pattern 3: Modify Metrics Based on Source

```
# Different metrics for different sources
set policy-options policy-statement METRIC-MODIFY term FROM-STATIC from protocol static
set policy-options policy-statement METRIC-MODIFY term FROM-STATIC then metric 100
set policy-options policy-statement METRIC-MODIFY term FROM-CONNECTED from protocol direct
set policy-options policy-statement METRIC-MODIFY term FROM-CONNECTED then metric 50
set policy-options policy-statement METRIC-MODIFY term DEFAULT then accept
```

Policy Evaluation Logic

The Decision Tree:

```
Start Policy Evaluation
    |
    v
Term 1: Match conditions?
    |
    +-- YES --> Execute actions
    |             |
    |             +-- accept/reject? --> DONE
    |             |
    |             +-- next term? --> Continue
    |
    +-- NO --> Next term
    |
    v
Term 2: Match conditions?
    |
    ... (continues)
    |
    v
End of Policy: No match?
    |
    v
Default Action (usually reject)
```

Troubleshooting Pattern #4: Policy Issues

1. Routes Not Being Accepted:

```
# Check if policy is applied
show configuration protocols ospf export

# Test policy without applying
test policy <POLICY-NAME> <ROUTE>

# Trace policy evaluation
show route protocol ospf detail
# Look for "Policy" in output
```

2. Wrong Routes Being Advertised:

```
# See what's being advertised
show route advertising-protocol ospf

# Check policy logic
show configuration policy-options policy-statement <POLICY-NAME>

# Common issue: Missing reject term at end
set policy-options policy-statement <POLICY-NAME> term LAST then reject
```

3. Policy Not Working as Expected:

```
# Enable policy tracing
set policy-options policy-statement <POLICY-NAME> term <TERM> then trace

# Check trace files
show log messages | match POLICY
```

Advanced Policy Features

1. Prefix Lists (Reusable Route Filters):

```
# Define prefix list
set policy-options prefix-list INTERNAL-NETS 10.0.0.0/8
set policy-options prefix-list INTERNAL-NETS 172.16.0.0/12
set policy-options prefix-list INTERNAL-NETS 192.168.0.0/16

# Use in policy
set policy-options policy-statement USE-PREFIX-LIST term 1 from prefix-list INTERNAL-NETS
set policy-options policy-statement USE-PREFIX-LIST term 1 then accept
```

2. Policy Subroutines:

```
# Create reusable policy
set policy-options policy-statement SET-METRIC-100 then metric 100

# Call from another policy
```

```
set policy-options policy-statement MAIN-POLICY term 1 from protocol static
set policy-options policy-statement MAIN-POLICY term 1 then policy SET-METRIC-100
```

3. Regular Expressions:

```
# Match AS paths (BGP)
set policy-options as-path TRANSIT-AS "65001 65002 .*"
set policy-options policy-statement BGP-FILTER term 1 from as-path TRANSIT-AS
```

Building Policy Instincts

Think in These Patterns:

1. **Source-Action Pattern:**
   - "From protocol X, do Y"
   - "From network A, set attribute B"
2. **Filter Pattern:**
   - "Accept these specific routes"
   - "Reject everything else"
3. **Modification Pattern:**
   - "Change metric for load balancing"
   - "Set communities for BGP control"

Common Mistakes to Avoid:

1. **Missing Default Action:**

   ```
   # Bad: No explicit final action
   # Good: Always end with explicit accept/reject
   set policy-options policy-statement MY-POLICY term LAST then reject
   ```

2. **Order Matters:**

   ```
   # Specific matches BEFORE general matches
   term 1: from route-filter 10.1.1.0/24 exact then reject
   term 2: from route-filter 10.0.0.0/8 orlonger then accept
   ```

3. **Import vs Export Confusion:**
   - Remember: Export = OUT of protocol
   - Import = INTO routing table from protocol

Policy Design Methodology

Step 1: Define Requirements

- What routes should be advertised?
- What routes should be filtered?
- What attributes need modification?

Step 2: Create Match Criteria

- By protocol?
- By network?
- By attributes?

Step 3: Define Actions

- Accept/Reject
- Modify attributes
- Add tags/communities

Step 4: Test Before Applying

```
test policy POLICY-NAME 10.1.1.0/24
```

Real-World Policy Example

Scenario: ISP Connection with Default Route

```
# Policy to advertise only our public prefix
set policy-options policy-statement TO-ISP term OUR-PREFIX from route-filter 203.0.113.0/24 exact
set policy-options policy-statement TO-ISP term OUR-PREFIX then accept
set policy-options policy-statement TO-ISP term DENY-ALL then reject

# Policy to accept only default route from ISP
set policy-options policy-statement FROM-ISP term DEFAULT from route-filter 0.0.0.0/0 exact
set policy-options policy-statement FROM-ISP term DEFAULT then accept
set policy-options policy-statement FROM-ISP term DENY-ALL then reject

# Apply to BGP
set protocols bgp group ISP neighbor 198.51.100.1 import FROM-ISP
set protocols bgp group ISP neighbor 198.51.100.1 export TO-ISP
```

Module 5 Preview: Deploying OSPF

Now that you understand routing policies, we'll apply this knowledge to real OSPF deployments. You'll learn to combine OSPF with policies for sophisticated routing control.

Module 5: Deploying OSPF

Moving from Theory to Practice

Now we combine OSPF fundamentals with routing policies to build production networks. Think of this as moving from knowing how to drive a car to actually navigating through city traffic.

OSPF Deployment Checklist

Before configuring OSPF, answer these questions:

1. **Network Design:**
    - How many routers?
    - What's the topology?
    - Where are the redundant paths?
2. **Area Design:**
    - Single area or multiple?
    - Where are area boundaries?
    - Which routers will be ABRs?
3. **IP Addressing:**
    - What's the addressing scheme?
    - Where to use loopbacks?
    - How to summarize?

The Complete OSPF Configuration Pattern

```
# Step 1: Router ID (Best Practice: Use Loopback)
set interfaces lo0 unit 0 family inet address 10.0.0.1/32
set routing-options router-id 10.0.0.1

# Step 2: OSPF Basic Configuration
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0

# Step 3: Export Policy (Redistribute Connected)
set policy-options policy-statement CONNECTED-TO-OSPF term 1 from protocol direct
set policy-options policy-statement CONNECTED-TO-OSPF term 1 from route-filter 192.168.0.0/16 orlonger
set policy-options policy-statement CONNECTED-TO-OSPF term 1 then accept
set protocols ospf export CONNECTED-TO-OSPF
```

Real-World Scenario 1: Enterprise Campus Network

The Topology:

```
         [Core-1]--------[Core-2]
         /      \        /      \
        /        \      /        \
    [Dist-1]    [Dist-2]    [Dist-3]
       |           |           |
    [Access]    [Access]    [Access]
```

Design Decisions:

- Core routers in Area 0
- Each distribution pair in separate area
- Summarize at distribution layer

Core Router Configuration:

```
# Core-1 Configuration
set interfaces lo0 unit 0 family inet address 10.0.0.1/32
set routing-options router-id 10.0.0.1

# Backbone Area
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 interface-type p2p  # To Core-2
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0 interface-type p2p  # To Dist-1
set protocols ospf area 0.0.0.0 interface ge-0/0/2.0 interface-type p2p  # To Dist-2

# Area 1 for Distribution Layer 1
set protocols ospf area 0.0.0.1 interface ge-0/0/3.0 interface-type p2p

# Enable Area Range (Summarization)
set protocols ospf area 0.0.0.1 area-range 10.1.0.0/16
```

Distribution Router Configuration:

```
# Dist-1 Configuration (ABR)
set interfaces lo0 unit 0 family inet address 10.1.0.1/32
set routing-options router-id 10.1.0.1

# Connection to Core (Area 0)
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 interface-type p2p
set protocols ospf area 0.0.0.0 interface lo0.0 passive

# Connection to Access (Area 1)
set protocols ospf area 0.0.0.1 interface ge-0/0/1.0
set protocols ospf area 0.0.0.1 interface vlan.100  # User VLAN
```

OSPF Network Types Deep Dive

1. Broadcast Networks (Ethernet Default)

```
# Default behavior - no configuration needed
# Elects DR/BDR
# Hello = 10s, Dead = 40s

# Force DR election
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 priority 200  # Higher = more likely DR
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0 priority 0    # Never become DR
```

2. Point-to-Point (Recommended for Router-to-Router)

```
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 interface-type p2p

# Benefits:
```

```
# - No DR election (faster convergence)
# - Only two routers expected
# - Simpler troubleshooting
```

3. NBMA (Non-Broadcast Multi-Access)

```
# For Frame Relay, ATM
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 interface-type nbma
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 neighbor 10.1.1.2
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 poll-interval 120
```

OSPF Authentication Deployment

Scenario: Securing OSPF in Production

```
# Method 1: Simple Password (Not Recommended)
set protocols ospf area 0.0.0.0 authentication-type simple
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 authentication-key MyPassword

# Method 2: MD5 (Recommended)
set protocols ospf area 0.0.0.0 authentication-type md5
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 authentication md5 1 key SecretKey123
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 authentication md5 2 key NewKey456 start-time 2024-12-01.00:00:00

# Key Rollover Pattern:
# 1. Add new key with future start-time
# 2. Old key still works until start-time
# 3. Graceful transition
```

Advanced OSPF Features

1. Virtual Links (Connecting to Backbone)

```
# When area not directly connected to Area 0
# Configure on both ABRs

# On ABR1 (Router ID: 10.0.0.1)
set protocols ospf area 0.0.0.1 virtual-link neighbor-id 10.0.0.2 transit-area 0.0.0.1

# On ABR2 (Router ID: 10.0.0.2)
set protocols ospf area 0.0.0.1 virtual-link neighbor-id 10.0.0.1 transit-area 0.0.0.1
```

2. Stub Areas (Reducing Database Size)

```
# Normal Stub Area (No External Routes)
set protocols ospf area 0.0.0.2 stub

# Totally Stubby Area (No External or Inter-Area)
set protocols ospf area 0.0.0.2 stub no-summaries

# Not-So-Stubby Area (NSSA) - Can originate external routes
set protocols ospf area 0.0.0.3 nssa
```

3. Route Redistribution with Control

```
# Sophisticated redistribution policy
set policy-options policy-statement REDIS-STATIC term CUSTOMER-ROUTES from protocol static
set policy-options policy-statement REDIS-STATIC term CUSTOMER-ROUTES from tag 100
set policy-options policy-statement REDIS-STATIC term CUSTOMER-ROUTES then {
    metric 100;
    external-type 1;  # Type 1 = metric increases
    accept;
}
set policy-options policy-statement REDIS-STATIC term REJECT then reject

set protocols ospf export REDIS-STATIC
```

OSPF Troubleshooting Methodology

Layer 1: Physical Connectivity

```
show interfaces ge-0/0/0 terse
show interfaces ge-0/0/0 extensive | match "error|drop"
ping 10.1.1.2 rapid count 5
```

Layer 2: OSPF Protocol

```
# Neighbor issues
show ospf neighbor
show ospf interface detail
show ospf statistics

# Real-time debugging
monitor traffic interface ge-0/0/0.0 matching "proto 89" write-file ospf.pcap
```

Layer 3: Database and Routes

```
# Database synchronization
show ospf database
show ospf database router detail
show ospf database network detail

# SPF calculation
show ospf route
show ospf spf log
```

Common OSPF Problems and Solutions

Problem 1: Neighbors Stuck in ExStart

```
# Diagnosis
show ospf neighbor
# State shows "ExStart"

# Common Causes:
# 1. MTU mismatch
show interfaces ge-0/0/0 | match MTU
set interfaces ge-0/0/0 mtu 1500

# 2. Duplicate Router ID
show ospf overview | match "Router ID"
set routing-options router-id 10.0.0.99
```

Problem 2: Routes Not Propagating

```
# Check 1: Is route in OSPF database?
show ospf database external extensive

# Check 2: Is it being filtered?
show configuration protocols ospf export

# Check 3: Area type restrictions?
show ospf area
# Stub areas don't accept external routes
```

Problem 3: Suboptimal Routing

```
# Check metrics
show ospf route 192.168.1.0 detail

# Adjust interface costs
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 metric 100
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0 metric 50
```

```
# Verify new path
show route 192.168.1.0 detail
```

OSPF Convergence Optimization

1. Faster Hello/Dead Timers

```
# Subsecond convergence
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 hello-interval 1
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 dead-interval 3
```

2. BFD Integration

```
# Bidirectional Forwarding Detection
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 bfd-liveness-detection minimum-interval 100
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 bfd-liveness-detection multiplier 3
```

3. SPF Throttling

```
# Control SPF calculations
set protocols ospf spf-options delay 200
set protocols ospf spf-options holddown 1000
set protocols ospf spf-options rapid-runs 5
```

OSPF Monitoring and Maintenance

Regular Health Checks:

```
# Daily checks
show ospf neighbor | match -v Full  # Any non-Full neighbors?
show ospf interface | match Dead     # Check dead timers
show ospf statistics | match Error   # Protocol errors

# Create monitoring script
show ospf route | count
# Save count, alert if changes dramatically
```

Maintenance Mode Pattern:

```
# Gracefully remove router from production
set protocols ospf overload

# Traffic diverts around this router
# Router still participates in OSPF
# Remove when maintenance complete:
delete protocols ospf overload
```

Lab Exercise Approach

Systematic Lab Methodology:

1. **Draw the Topology First**
   - Mark areas
   - Note IP addresses
   - Identify ABRs
2. **Configure in Order:**
   - Physical interfaces
   - Loopbacks and Router IDs
   - OSPF areas
   - Policies
   - Verification
3. **Verify Each Step:**

```
show ospf neighbor
show ospf interface
show route protocol ospf
```

4. **Test Failover:**

```
# Disable primary path
set interfaces ge-0/0/0 disable
commit

# Verify backup path active
show route 192.168.1.0
```

Building OSPF Instincts

Pattern Recognition:

- See "Init" → Think MTU
- See "ExStart" → Think Router ID
- See "Loading" → Normal, just wait
- See no neighbors → Check interface configuration

Design Patterns:

- Hub sites = Area 0
- Remote sites = Non-zero areas
- WAN links = Point-to-point
- Campus = Area per building

Troubleshooting Reflex:

1. Physical up? (`show interfaces terse`)
2. OSPF enabled? (`show ospf interface`)
3. Neighbors up? (`show ospf neighbor`)
4. Routes learned? (`show route protocol ospf`)

Module 6 Preview: IS-IS

While OSPF uses areas for hierarchy, IS-IS uses levels. We'll explore this alternative link-state protocol and when to choose IS-IS over OSPF.

Module 6: IS-IS (Intermediate System to Intermediate System)

What is IS-IS? - The Alternative Link-State Protocol

IS-IS is like OSPF's cousin from a different family. While OSPF was built specifically for IP, IS-IS was designed to be protocol-independent. Think of it as a universal translator that happened to become very good at speaking IP.

Historical Context:

```
OSI Model Intent:        Reality Today:
- CLNP Routing          - IP Routing only
- Protocol neutral      - Simpler than OSPF
- Telco heritage        - Popular in ISP cores
```

IS-IS vs OSPF - The Key Differences

Fundamental Differences:

```
OSPF:                           IS-IS:
- Runs over IP (Protocol 89)    - Runs directly over Layer 2
- Areas change at ABRs          - Levels overlap on same router
- Complex packet types          - Simple, extensible TLVs
- Router ID = IP address        - System ID = MAC-like address
```

IS-IS Addressing - Understanding NETs

The Network Entity Title (NET) is IS-IS's version of a Router ID, but more complex:

```
NET Example: 49.0001.0100.0000.0001.00

Breaking it down:
49.0001   - Area ID (like OSPF area, but in hex)
0100.0000.0001 - System ID (unique router identifier)
00        - NSEL (always 00 for routers)
```

Think of it like a phone number:

- Area ID = Country/City code
- System ID = Your unique number
- NSEL = Extension (00 for routers)

IS-IS Levels - The Hierarchy

Unlike OSPF's strict area boundaries, IS-IS uses overlapping levels:

```
Level 1: Intra-area routing (like OSPF internal routes)
Level 2: Inter-area routing (like OSPF backbone)
Level 1/2: Both (default for Junos)

Visual Representation:
[L1/L2 Router] <--L2 Only--> [L1/L2 Router]
      |                            |
   L1 Only                      L1 Only
      |                            |
[L1 Router]                   [L1 Router]
```

IS-IS PDUs - The Communication Types

IS-IS uses PDUs (Protocol Data Units) instead of packets:

1. **Hello PDUs** - For neighbor discovery
   - L1 LAN Hello
   - L2 LAN Hello
   - Point-to-point Hello
2. **LSP (Link State PDU)** - Database information
   - L1 LSP
   - L2 LSP
3. **CSNP (Complete Sequence Number PDU)** - Database summary
   - Like OSPF Database Description
4. **PSNP (Partial Sequence Number PDU)** - Request specific info
   - Like OSPF LS Request

Junos CLI Pattern #9: Basic IS-IS Configuration

```
# Step 1: Configure NET on loopback
set interfaces lo0 unit 0 family iso address 49.0001.0100.0000.0001.00

# Step 2: Enable IS-IS
set protocols isis interface ge-0/0/0.0
set protocols isis interface lo0.0

# The pattern breakdown:
# - No area configuration needed (it's in the NET)
# - Simpler than OSPF
# - Level 1/2 by default
```

Complete Basic Configuration:

```
# Router 1
set interfaces lo0 unit 0 family iso address 49.0001.0100.0000.0001.00
set interfaces ge-0/0/0 unit 0 family iso
set protocols isis interface ge-0/0/0.0
set protocols isis interface lo0.0
```

Understanding IS-IS Levels in Practice

Configuring Level Types:

```
# Level 1 only (edge router)
set protocols isis interface ge-0/0/0.0 level 2 disable

# Level 2 only (backbone router)
set protocols isis interface ge-0/0/0.0 level 1 disable

# Level 1/2 (default - border router)
# No configuration needed
```

Level Interaction Rules:

- L1 routers only talk L1 to each other
- L2 routers only talk L2 to each other
- L1/L2 routers adapt based on neighbor

IS-IS Adjacency Formation

The Adjacency State Machine:

```
Down -> Init -> Up

Simpler than OSPF!

Down: No hellos received
Init: One-way communication
Up: Two-way communication established
```

Adjacency Requirements:

- Same level capability
- Authentication match (if configured)
- MTU compatibility
- Area match (for L1 adjacencies)

DIS Election - The IS-IS DR

DIS (Designated IS) is IS-IS's version of OSPF's DR, but simpler:

Key Differences from OSPF DR:

```
OSPF DR:                    IS-IS DIS:
- Has backup (BDR)         - No backup
- Complex election         - Simple priority
- Sticky (stays DR)        - Preemptive
- Special LSAs             - Regular LSPs
```

DIS Configuration:

```
# Set DIS priority (higher wins)
set protocols isis interface ge-0/0/0.0 level 1 priority 100
set protocols isis interface ge-0/0/0.0 level 2 priority 100

# Priority 0 = never become DIS (unlike OSPF)
```

Junos CLI Pattern #10: IS-IS Verification

```
# Verification commands
show isis adjacency            # Who are my neighbors?
show isis interface           # Which interfaces run IS-IS?
show isis database            # What's in the link-state database?
show isis route              # What routes did IS-IS calculate?
show route protocol isis     # What made it to routing table?
```

Reading IS-IS Adjacency Output:

```
Interface  System         L State  Hold (secs) SNPA
ge-0/0/0.0 Router2        2 Up     24          0:c:29:1:2:3

L = Level (1, 2, or 3 for both)
State = Up (what we want)
Hold = Holdtime remaining
SNPA = Subnetwork Point of Attachment (MAC address)
```

IS-IS Metrics and Path Selection

Metric Types:

```
Original (Narrow) Metrics: 1-63 per interface
Wide Metrics: 1-16777215 per interface (default in Junos)

# Always use wide metrics in production:
set protocols isis level 1 wide-metrics-only
set protocols isis level 2 wide-metrics-only
```

Setting Interface Metrics:

```
# Set metric for specific level
set protocols isis interface ge-0/0/0.0 level 1 metric 100
set protocols isis interface ge-0/0/0.0 level 2 metric 50

# Different metrics for different levels allows traffic engineering
```

IS-IS Authentication

```
# Simple authentication (not recommended)
set protocols isis interface ge-0/0/0.0 hello-authentication-key SecretKey
set protocols isis interface ge-0/0/0.0 hello-authentication-type simple

# MD5 authentication (recommended)
set protocols isis interface ge-0/0/0.0 hello-authentication-key "$9$encrypted"
set protocols isis interface ge-0/0/0.0 hello-authentication-type md5

# LSP authentication (database protection)
set protocols isis level 1 authentication-key "$9$encrypted"
set protocols isis level 1 authentication-type md5
```

Advanced IS-IS Features

1. Route Leaking (L2 to L1)

```
# Define policy to leak specific routes
set policy-options policy-statement L2-TO-L1 term 1 from protocol isis
set policy-options policy-statement L2-TO-L1 term 1 from level 2
set policy-options policy-statement L2-TO-L1 term 1 from route-filter 10.0.0.0/8 longer
set policy-options policy-statement L2-TO-L1 term 1 to level 1
set policy-options policy-statement L2-TO-L1 term 1 then accept
```

```
# Apply the policy
set protocols isis export L2-TO-L1
```

2. IS-IS Overload Bit

```
# Set overload (removes router from transit)
set protocols isis overload

# With timeout
set protocols isis overload timeout 300  # 5 minutes

# Advertise own prefixes only
set protocols isis overload advertise-high-metrics
```

3. BFD with IS-IS

```
# Enable BFD for fast failure detection
set protocols isis interface ge-0/0/0.0 bfd-liveness-detection minimum-interval 100
set protocols isis interface ge-0/0/0.0 bfd-liveness-detection multiplier 3
```

Troubleshooting Pattern #5: IS-IS Issues

Problem 1: No Adjacency

```
# Check 1: ISO family on interface?
show interfaces ge-0/0/0.0 | match iso

# Fix if missing:
set interfaces ge-0/0/0 unit 0 family iso

# Check 2: IS-IS enabled on interface?
show isis interface

# Check 3: Matching levels?
show isis interface detail
```

Problem 2: Adjacency Up but No Routes

```
# Check database
show isis database detail

# Check route calculation
show isis route

# Check if routes installed
show route protocol isis

# Common issue: Level mismatch
# L1 router won't see L2 routes without route leaking
```

Problem 3: NET Configuration Errors

```
# Verify NET
show isis hostname

# Common NET mistakes:
# - Wrong area (first part)
# - Duplicate system ID (middle part)
# - NSEL not 00 (end part)

# Fix:
set interfaces lo0 unit 0 family iso address 49.0001.0100.0000.9999.00
```

When to Use IS-IS vs OSPF

Choose IS-IS When:

- Running a large ISP network
- Need simpler protocol operation
- Want better scaling characteristics
- Migrating from legacy OSI networks

Choose OSPF When:

- Enterprise environment
- Staff familiar with OSPF
- Need virtual links
- Want more granular area types

IS-IS Design Patterns

Pattern 1: ISP Backbone

```
All routers: Level 2 only
Simple, scalable, no area planning needed

set protocols isis interface all level 1 disable
```

Pattern 2: Enterprise with Hierarchy

```
Core: Level 2 only
Distribution: Level 1/2
Access: Level 1 only

Creates natural hierarchy without complex area planning
```

Pattern 3: Route Leaking for Optimal Routing

```
# Leak specific prefixes from L2 to L1
set policy-options policy-statement LEAK-SPECIFICS from route-filter 0.0.0.0/0 exact
set policy-options policy-statement LEAK-SPECIFICS to level 1
set policy-options policy-statement LEAK-SPECIFICS then accept
set protocols isis export LEAK-SPECIFICS
```

Building IS-IS Instincts

Quick Diagnostics:

- No adjacency? → Check ISO family
- Wrong level adjacency? → Check interface levels
- No routes? → Check database and levels
- Suboptimal routing? → Consider route leaking

Configuration Reflexes:

1. Always configure ISO on lo0 first
2. Enable family ISO on interfaces
3. Add interfaces to IS-IS
4. Verify with `show isis adjacency`

Design Principles:

- Simpler is better (use single level when possible)
- Wide metrics always
- BFD for fast convergence
- Route leaking for optimal routing

Real-World IS-IS Example

Scenario: ISP Core Network

```
# Core Router Configuration
set interfaces lo0 unit 0 family iso address 49.0001.0100.0000.0001.00
set interfaces lo0 unit 0 family inet address 10.0.0.1/32

# All interfaces Level 2 only
set protocols isis interface all level 1 disable
set protocols isis interface ge-0/0/0.0
set protocols isis interface ge-0/0/1.0
set protocols isis interface lo0.0

# Wide metrics
set protocols isis level 2 wide-metrics-only

# Authentication
set protocols isis level 2 authentication-key "$9$secret"
set protocols isis level 2 authentication-type md5

# BFD on all interfaces
set protocols isis interface ge-0/0/0.0 bfd-liveness-detection minimum-interval 100
set protocols isis interface ge-0/0/1.0 bfd-liveness-detection minimum-interval 100

# Reference bandwidth for auto-costing
set protocols isis reference-bandwidth 100g
```

Module 7 Preview: Fundamentals of BGP

Now we move from interior protocols (OSPF/IS-IS) to the protocol that runs the Internet - BGP. This is where routing becomes more about business relationships than shortest paths.

Module 7: Fundamentals of BGP (Border Gateway Protocol)

What is BGP? - The Internet's Routing Protocol

BGP is fundamentally different from OSPF and IS-IS. While those protocols find the shortest path, BGP is about business relationships and policy. Think of it as international trade routes versus local city roads.

The Key Paradigm Shift:

```
Interior Protocols (OSPF/IS-IS):        BGP:
- Find shortest path                    - Follow business agreements
- All routers trust each other         - Trust no one fully
- Automatic best path                  - Policy drives decisions
- Fast convergence                     - Stability over speed
```

BGP's Role in the Internet

```
Your Company AS 65001
        |
        | (Customer-Provider Relationship)
        |
   ISP-A AS 701 ------------ ISP-B AS 702
        |     (Peer Relationship)      |
        |                              |
   Other Customers              Other Customers
```

Understanding Autonomous Systems (AS)

An AS is like a country in the Internet world - it has its own internal rules but must cooperate with others for global connectivity.

AS Number Ranges:

```
1-64495:        Public AS numbers (globally unique)
64496-64511:    Reserved for documentation
64512-65534:    Private AS numbers (like RFC1918)
```

```
65535:          Reserved
4200000000+:    32-bit AS numbers
```

BGP Attributes - The Route Properties

BGP doesn't use simple metrics. Instead, it uses multiple attributes that form a complex decision process:

Key Attributes (in decision order):

```
1. Next-Hop        - Must be reachable
2. Weight          - Cisco proprietary (not in Junos)
3. Local Preference - Internal tiebreaker (higher wins)
4. AS Path         - List of ASes (shorter wins)
5. Origin          - IGP < EGP < Incomplete
6. MED             - Suggestion from neighbor (lower wins)
7. eBGP vs iBGP    - External preferred
8. IGP Metric      - Distance to next-hop
```

BGP Message Types

BGP uses TCP port 179 for reliable communication:

1. **OPEN** - Establish session
2. **UPDATE** - Advertise/withdraw routes
3. **KEEPALIVE** - Maintain session
4. **NOTIFICATION** - Error message

eBGP vs iBGP - External vs Internal

This is crucial to understand:

```
eBGP (External BGP):           iBGP (Internal BGP):
- Between different ASes       - Within same AS
- TTL=1 by default            - TTL=255
- Next-hop changes            - Next-hop unchanged
- AS-Path updated             - AS-Path unchanged
- Routes freely advertised    - Routes not re-advertised
```

The iBGP Full Mesh Rule:

```
In AS 65001:
[R1]----[R2]
 | \    / |
 |  \  /  |
 |   \/   |
 |   /\   |
 |  /  \  |
 | /    \ |
[R3]----[R4]

Every router must peer with every other router!
(Or use Route Reflectors/Confederations)
```

Junos CLI Pattern #11: Basic BGP Configuration

```
# Step 1: Define local AS
set routing-options autonomous-system 65001

# Step 2: Create BGP group (logical grouping)
set protocols bgp group EXTERNAL type external

# Step 3: Define neighbor
set protocols bgp group EXTERNAL neighbor 192.168.1.1 peer-as 65002
```

```
# The pattern breakdown:
# - autonomous-system: Define your AS once globally
# - group: Logical container for similar neighbors
# - type: external or internal
# - neighbor: Specific peer configuration
# - peer-as: Remote AS (required for eBGP)
```

Complete eBGP Configuration Example

```
# Local AS
set routing-options autonomous-system 65001

# eBGP to ISP
set protocols bgp group ISP-A type external
set protocols bgp group ISP-A neighbor 198.51.100.1 peer-as 701
set protocols bgp group ISP-A neighbor 198.51.100.1 description "ISP-A Primary Link"

# What we accept (import policy)
set policy-options policy-statement FROM-ISP term ACCEPT-DEFAULT from route-filter 0.0.0.0/0 exact
set policy-options policy-statement FROM-ISP term ACCEPT-DEFAULT then accept
set policy-options policy-statement FROM-ISP term DENY-REST then reject

# What we advertise (export policy)
set policy-options policy-statement TO-ISP term OUR-PREFIXES from route-filter 203.0.113.0/24 exact
set policy-options policy-statement TO-ISP term OUR-PREFIXES then accept
set policy-options policy-statement TO-ISP term DENY-REST then reject

# Apply policies
set protocols bgp group ISP-A neighbor 198.51.100.1 import FROM-ISP
set protocols bgp group ISP-A neighbor 198.51.100.1 export TO-ISP
```

Complete iBGP Configuration Example

```
# iBGP full mesh within AS 65001
set routing-options autonomous-system 65001

# iBGP group
set protocols bgp group INTERNAL type internal
set protocols bgp group INTERNAL local-address 10.0.0.1
set protocols bgp group INTERNAL neighbor 10.0.0.2
set protocols bgp group INTERNAL neighbor 10.0.0.3
set protocols bgp group INTERNAL neighbor 10.0.0.4

# iBGP specific settings
set protocols bgp group INTERNAL next-hop-self
# Changes next-hop to self for eBGP learned routes
```

Understanding BGP Path Selection

The Decision Process (Simplified):

```
Routes from neighbors
       |
    Valid?  ----No----> Discard
       |
      Yes
       |
    Apply import policy
       |
    BGP Decision Process
       |
    Best path selected
       |
    Apply export policy
```

```
         |
     Advertise to neighbors
```

Junos CLI Pattern #12: BGP Verification

```
# BGP session status
show bgp summary

# Detailed neighbor information
show bgp neighbor 192.168.1.1

# Routes received from neighbor
show route receive-protocol bgp 192.168.1.1

# Routes advertised to neighbor
show route advertising-protocol bgp 192.168.1.1

# BGP-specific route details
show route protocol bgp detail
```

Reading BGP Summary Output:

```
Peer              AS      InPkt    OutPkt   State|#Active/Received/Accepted/Damped...
192.168.1.1    65002       1234      1235   Established 10/15/12/0 0/0/0/0
                                             ^^^^^^^^^^^
                                             This is what we want!

States:
- Idle: No attempt to connect
- Connect: TCP connection attempt
- Active: TCP failed, trying again
- OpenSent: TCP up, OPEN sent
- OpenConfirm: OPEN received, waiting for KEEPALIVE
- Established: Full operation
```

BGP Attributes Deep Dive

1. Local Preference (iBGP only)

```
# Set local preference (default 100)
set policy-options policy-statement SET-LOCAL-PREF term 1 from neighbor 192.168.1.1
set policy-options policy-statement SET-LOCAL-PREF term 1 then local-preference 200
set policy-options policy-statement SET-LOCAL-PREF term 1 then accept

# Higher local preference wins!
```

2. AS Path Manipulation

```
# Prepend AS for traffic engineering
set policy-options policy-statement PREPEND term 1 then as-path-prepend "65001 65001"
# Makes path look longer, less preferred

# AS Path regex matching
set policy-options as-path TRANSIT ".* 701 .*"
set policy-options policy-statement MATCH-TRANSIT from as-path TRANSIT
```

3. MED (Multi-Exit Discriminator)

```
# Suggest entry point to neighbor AS
set policy-options policy-statement SET-MED term 1 then metric 100
# Lower MED preferred (like IGP metrics)
```

BGP Communities - Route Tagging

Communities are like luggage tags for routes:

```
# Well-known communities
NO_EXPORT      = 65535:65281  # Don't advertise outside AS
NO_ADVERTISE   = 65535:65282  # Don't advertise to any peer
NO_EXPORT_SUBCONFED = 65535:65283

# Define custom communities
set policy-options community CUSTOMER members 65001:100
set policy-options community TRANSIT members 65001:200

# Tag routes
set policy-options policy-statement TAG-CUSTOMER from protocol direct
set policy-options policy-statement TAG-CUSTOMER then community add CUSTOMER
```

BGP Troubleshooting Pattern

Problem 1: Session Won't Establish

```
# Check 1: TCP connectivity
ping 192.168.1.1
telnet 192.168.1.1 port 179

# Check 2: Configuration
show configuration protocols bgp group EXTERNAL

# Check 3: AS numbers correct?
show bgp neighbor 192.168.1.1 | match "Peer AS|Local AS"

# Check 4: For eBGP, TTL issues?
set protocols bgp group EXTERNAL neighbor 192.168.1.1 multihop ttl 2
```

Problem 2: Routes Not Received

```
# Check what neighbor sends
show route receive-protocol bgp 192.168.1.1 hidden

# Check import policy
show configuration protocols bgp group EXTERNAL neighbor 192.168.1.1 import

# Test policy
test policy FROM-ISP 0.0.0.0/0
```

Problem 3: Routes Not Advertised

```
# Check if route exists
show route 203.0.113.0/24

# Check export policy
show configuration protocols bgp group EXTERNAL neighbor 192.168.1.1 export

# Verify what's advertised
show route advertising-protocol bgp 192.168.1.1
```

BGP Best Practices

1. Always Use Policies

```
# Never use BGP without explicit policies
# Default Junos behavior:
# - eBGP: Accept all, advertise nothing
# - iBGP: Accept all, advertise all BGP routes

# Minimum safe configuration:
set policy-options policy-statement DENY-ALL then reject
set protocols bgp group EXTERNAL neighbor 192.168.1.1 import SPECIFIC-POLICY
set protocols bgp group EXTERNAL neighbor 192.168.1.1 export SPECIFIC-POLICY
```

2. Prefix Limits

```
# Prevent neighbor from filling routing table
set protocols bgp group EXTERNAL neighbor 192.168.1.1 family inet unicast prefix-limit maximum 1000
set protocols bgp group EXTERNAL neighbor 192.168.1.1 family inet unicast prefix-limit teardown 90
```

3. Authentication

```
# MD5 authentication
set protocols bgp group EXTERNAL neighbor 192.168.1.1 authentication-key "$9$secret"
```

Common BGP Scenarios

Scenario 1: Dual ISP with Primary/Backup

```
# Prefer ISP-A (higher local preference)
set policy-options policy-statement FROM-ISP-A then local-preference 200
set policy-options policy-statement FROM-ISP-B then local-preference 100

# Advertise with prepend to ISP-B
set policy-options policy-statement TO-ISP-B then as-path-prepend "65001 65001 65001"
```

Scenario 2: Customer Routes with Communities

```
# Tag customer routes
set policy-options policy-statement FROM-CUSTOMER then community add CUSTOMER
set policy-options policy-statement FROM-CUSTOMER then local-preference 150

# Don't export customer routes to other customers
set policy-options policy-statement TO-CUSTOMER from community CUSTOMER
set policy-options policy-statement TO-CUSTOMER then reject
```

Building BGP Instincts

State Machine Understanding:

- Idle → Connect: Configuration activated
- Active state: Usually connectivity issue
- Established → Active: Something broke

Policy Thinking:

- What should I accept?
- What should I advertise?
- How should I modify attributes?

Troubleshooting Reflex:

1. Session up? (`show bgp summary`)
2. Routes received? (`show route receive-protocol bgp`)
3. Routes accepted? (`show route protocol bgp`)
4. Routes advertised? (`show route advertising-protocol bgp`)

Real-World BGP Example

Enterprise Dual-Homed to ISPs:

```
# AS and router ID
set routing-options autonomous-system 65001
set routing-options router-id 10.0.0.1

# eBGP to ISP-A (primary)
set protocols bgp group ISP-A type external
set protocols bgp group ISP-A peer-as 701
set protocols bgp group ISP-A neighbor 198.51.100.1 description "ISP-A Primary"
set protocols bgp group ISP-A neighbor 198.51.100.1 import FROM-ISP-A
set protocols bgp group ISP-A neighbor 198.51.100.1 export TO-ISP-PRIMARY

# eBGP to ISP-B (backup)
set protocols bgp group ISP-B type external
set protocols bgp group ISP-B peer-as 702
set protocols bgp group ISP-B neighbor 198.51.100.5 description "ISP-B Backup"
set protocols bgp group ISP-B neighbor 198.51.100.5 import FROM-ISP-B
set protocols bgp group ISP-B neighbor 198.51.100.5 export TO-ISP-BACKUP

# Policies for primary/backup behavior
set policy-options policy-statement FROM-ISP-A term DEFAULT from route-filter 0.0.0.0/0 exact
set policy-options policy-statement FROM-ISP-A term DEFAULT then local-preference 200
set policy-options policy-statement FROM-ISP-A term DEFAULT then accept

set policy-options policy-statement FROM-ISP-B term DEFAULT from route-filter 0.0.0.0/0 exact
set policy-options policy-statement FROM-ISP-B term DEFAULT then local-preference 100
set policy-options policy-statement FROM-ISP-B term DEFAULT then accept

# Make backup path less attractive inbound
set policy-options policy-statement TO-ISP-BACKUP term PREPEND then as-path-prepend "65001 65001"
```

Module 8 Preview: Deploying BGP

Now that you understand BGP fundamentals, we'll explore advanced deployment scenarios including route reflectors, confederations, and complex policy implementations.

Module 8: Deploying BGP

The iBGP Full Mesh Problem

As your network grows, the iBGP full mesh requirement becomes unmanageable:

```
4 routers = 6 sessions
10 routers = 45 sessions
20 routers = 190 sessions
n routers = n(n-1)/2 sessions

Visual representation of the problem:
[R1]──────────[R2]
 | \        / |
 |  \      /  |
 |   \ /      |
 |   / \      |
 |  /    \    |
 | /       \  |
[R3]──────────[R4]
```

Solution 1: Route Reflection

Route Reflectors (RR) break the iBGP rule by re-advertising iBGP routes to other iBGP peers:

```
Traditional Full Mesh:          With Route Reflector:
    [R1]---[R2]                      [R1]
     | \  / |                         |
     |  \/  |                        [RR]
     |  /\  |                       / | \
     | /  \ |                      /  |   \
    [R3]---[R4]                 [R2]  [R3]  [R4]

6 sessions needed              4 sessions needed
```

Route Reflector Concepts

Three Types of Peers from RR Perspective:

1. **eBGP peers** - Normal eBGP rules
2. **Client peers** - RR reflects routes TO and FROM them
3. **Non-client peers** - Normal iBGP peers (must be fully meshed)

Reflection Rules:

```
Route from eBGP     → Reflect to all iBGP (clients and non-clients)
Route from client   → Reflect to all except originating client
Route from non-client → Reflect to clients only
```

Junos CLI Pattern #13: Route Reflector Configuration

```
# On the Route Reflector:
set protocols bgp group RR-CLIENTS type internal
set protocols bgp group RR-CLIENTS local-address 10.0.0.1
set protocols bgp group RR-CLIENTS cluster 10.0.0.1
set protocols bgp group RR-CLIENTS neighbor 10.0.0.2
set protocols bgp group RR-CLIENTS neighbor 10.0.0.3
set protocols bgp group RR-CLIENTS neighbor 10.0.0.4

# The key command that makes it a route reflector:
# "cluster" defines the cluster ID and enables reflection
```

On the Clients (Simple!):

```
# Clients just peer with RR normally
set protocols bgp group INTERNAL type internal
set protocols bgp group INTERNAL local-address 10.0.0.2
set protocols bgp group INTERNAL neighbor 10.0.0.1  # The RR
```

Route Reflector Redundancy

Always deploy RRs in pairs for redundancy:

```
     [RR1]----[RR2]
      / \      / \
    /    \   /    \
  /       \ /       \
[C1]      [C2]      [C3]
Each client peers with both RRs
```

Redundant RR Configuration:

```
# RR1 configuration
set protocols bgp group RR-CLIENTS type internal
set protocols bgp group RR-CLIENTS local-address 10.0.0.1
set protocols bgp group RR-CLIENTS cluster 1.1.1.1  # Same cluster ID!
set protocols bgp group RR-CLIENTS neighbor 10.0.0.11  # Client 1
set protocols bgp group RR-CLIENTS neighbor 10.0.0.12  # Client 2

# RR2 configuration (identical cluster ID)
set protocols bgp group RR-CLIENTS type internal
set protocols bgp group RR-CLIENTS local-address 10.0.0.2
set protocols bgp group RR-CLIENTS cluster 1.1.1.1  # Same as RR1!
set protocols bgp group RR-CLIENTS neighbor 10.0.0.11
set protocols bgp group RR-CLIENTS neighbor 10.0.0.12
```

BGP Route Reflection Attributes

Route Reflectors add attributes to prevent loops:

ORIGINATOR_ID: Router ID of route originator **CLUSTER_LIST**: List of cluster IDs (like AS_PATH for RRs)

```
# View these attributes:
show route protocol bgp detail
```

```
# Output includes:
#   Originator ID: 10.0.0.5
#   Cluster list:  1.1.1.1
```

Solution 2: BGP Confederations

Confederations divide a large AS into smaller sub-ASes:

```
Public View:                 Internal View:

AS 65001                    [Sub-AS 65101]--[Sub-AS 65102]
                                  |               |
                            [Sub-AS 65103]--[Sub-AS 65104]
```

Confederation Configuration

```
# Global confederation settings
set routing-options autonomous-system 65001           # Public AS
set routing-options confederation 65001 members [ 65101 65102 65103 ]

# BGP configuration treats sub-AS as external
set protocols bgp group CONFED-PEER type external
set protocols bgp group CONFED-PEER peer-as 65102    # Neighbor's sub-AS
set protocols bgp group CONFED-PEER neighbor 10.1.1.2
```

Route Reflector vs Confederation

Choose Route Reflector when:

- Simpler configuration needed
- Most networks use this
- Easy to migrate existing network
- Better vendor support

Choose Confederation when:

- Clear administrative boundaries exist
- Want AS_PATH loop prevention internally
- Complex TE requirements
- Historical/political reasons

Advanced BGP Deployment Patterns

Pattern 1: Hierarchical Route Reflection

```
         [Core-RR1]------[Core-RR2]
         /    |    \    /    |    \
       /      |      \/      |      \
     /        |      /\      |        \
[Regional] [Regional]  [Regional] [Regional]
[RR-West]  [RR-East]   [RR-North] [RR-South]
    |          |           |          |
[Clients]  [Clients]   [Clients]  [Clients]
```

Configuration for Hierarchical RR:

```
# Core RR peers with Regional RRs as clients
set protocols bgp group REGIONAL-RRS type internal
set protocols bgp group REGIONAL-RRS cluster 0.0.0.1
set protocols bgp group REGIONAL-RRS neighbor 10.1.0.1  # Regional-West
set protocols bgp group REGIONAL-RRS neighbor 10.2.0.1  # Regional-East

# Regional RR has two groups
# Upward to Core RRs (normal iBGP)
set protocols bgp group CORE-RRS type internal
set protocols bgp group CORE-RRS neighbor 10.0.0.1
set protocols bgp group CORE-RRS neighbor 10.0.0.2

# Downward to clients (as RR)
set protocols bgp group EDGE-CLIENTS type internal
set protocols bgp group EDGE-CLIENTS cluster 10.1.0.1
set protocols bgp group EDGE-CLIENTS neighbor 10.1.1.1
```

BGP ADD-PATH - Multiple Path Advertisement

Traditional BGP advertises only the best path. ADD-PATH allows advertising multiple paths:

```
# Enable ADD-PATH
set protocols bgp group RR-CLIENTS family inet unicast add-path send path-count 4
set protocols bgp group RR-CLIENTS neighbor 10.0.0.2 family inet unicast add-path receive

# Benefits:
# - Faster convergence
# - Better load balancing
# - Improved path diversity
```

BGP Security Features

1. GTSM (Generalized TTL Security Mechanism)

```
# Protects against remote attacks
set protocols bgp group EXTERNAL neighbor 192.168.1.1 ttl 255
# Only accepts packets with TTL=255 (must be directly connected)
```

2. TCP-AO (TCP Authentication Option)

```
# Stronger than MD5
set protocols bgp group EXTERNAL neighbor 192.168.1.1 authentication-algorithm ao
set protocols bgp group EXTERNAL neighbor 192.168.1.1 authentication-key-chain BGP-KEYS
```

3. Prefix Validation (RPKI)

```
# Validate routes against RPKI
set routing-options validation group RPKI session 10.0.0.100 port 323
set policy-options policy-statement RPKI-VALIDATE term VALID from validation-database valid
set policy-options policy-statement RPKI-VALIDATE term VALID then local-preference 110
```

Troubleshooting BGP at Scale

Problem 1: Route Reflector Client Not Receiving Routes

```
# On RR, check if reflecting:
show route advertising-protocol bgp 10.0.0.2  # To client

# Check cluster configuration:
show bgp group RR-CLIENTS

# Verify client is configured as RR client:
show configuration protocols bgp group RR-CLIENTS neighbor 10.0.0.2
```

Problem 2: Routing Loops with Multiple RRs

```
# Check for different cluster IDs (bad!):
show route protocol bgp detail | match "Cluster list"

# Solution: Ensure all RRs in same cluster use same ID
set protocols bgp group RR-CLIENTS cluster 1.1.1.1
```

Problem 3: Suboptimal Routing with RR

```
# RR doesn't modify next-hop by default
# Clients might have unreachable next-hops

# Solution 1: Next-hop-self on edge routers
set protocols bgp group INTERNAL next-hop-self

# Solution 2: IGP carries external next-hops
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 passive
```

BGP Convergence Optimization

1. Fast External Failover

```
# Immediately teardown eBGP if link fails
set protocols bgp group EXTERNAL neighbor 192.168.1.1 fast-external-failover
```

2. BFD Integration

```
# Subsecond failure detection
set protocols bgp group EXTERNAL neighbor 192.168.1.1 bfd-liveness-detection minimum-interval 300
set protocols bgp group EXTERNAL neighbor 192.168.1.1 bfd-liveness-detection multiplier 3
```

3. BGP PIC (Prefix Independent Convergence)

```
# Pre-calculate backup paths
set routing-options protect-core
set protocols bgp group INTERNAL family inet unicast protection
```

Real-World Deployment Example

Large Enterprise with Multiple Sites:

```
# Core Route Reflector Configuration
set routing-options router-id 10.0.0.1
set routing-options autonomous-system 65001

# iBGP RR clients (edge routers)
set protocols bgp group RR-CLIENTS type internal
set protocols bgp group RR-CLIENTS local-address 10.0.0.1
set protocols bgp group RR-CLIENTS cluster 0.0.0.1
set protocols bgp group RR-CLIENTS neighbor 10.1.0.1 description "Site-A Edge"
set protocols bgp group RR-CLIENTS neighbor 10.2.0.1 description "Site-B Edge"
set protocols bgp group RR-CLIENTS neighbor 10.3.0.1 description "Site-C Edge"

# iBGP to other RRs (non-client)
set protocols bgp group CORE-MESH type internal
set protocols bgp group CORE-MESH local-address 10.0.0.1
set protocols bgp group CORE-MESH neighbor 10.0.0.2 description "Core-RR2"

# Edge Router Configuration (RR Client)
set routing-options router-id 10.1.0.1
set routing-options autonomous-system 65001

# eBGP to ISP
set protocols bgp group ISP type external
set protocols bgp group ISP peer-as 701
set protocols bgp group ISP neighbor 198.51.100.1 description "ISP-A Circuit"
set protocols bgp group ISP neighbor 198.51.100.1 import FROM-ISP
set protocols bgp group ISP neighbor 198.51.100.1 export TO-ISP

# iBGP to Route Reflectors
set protocols bgp group INTERNAL type internal
set protocols bgp group INTERNAL local-address 10.1.0.1
set protocols bgp group INTERNAL neighbor 10.0.0.1 description "Core-RR1"
set protocols bgp group INTERNAL neighbor 10.0.0.2 description "Core-RR2"
set protocols bgp group INTERNAL next-hop-self
```

BGP Design Best Practices

1. RR Placement

- Place RRs out of data path
- Use dedicated RR routers when possible
- Always deploy in pairs

2. Cluster Design

- One cluster per POP/region
- Hierarchical for very large networks
- Same cluster ID for all RRs in cluster

3. Policy Application

- Apply policies at edge
- Keep RR configuration simple
- Use communities for policy signaling

Building Advanced BGP Instincts

Design Patterns:

- < 10 routers: Full mesh
- 10-50 routers: Single RR cluster
- 50+ routers: Hierarchical RR
- Clear boundaries: Consider confederation

Troubleshooting Reflexes:

- Missing routes? Check RR client configuration
- Loops? Check cluster IDs
- Suboptimal paths? Check next-hop reachability
- Slow convergence? Add BFD

Common Mistakes:

- Different cluster IDs on redundant RRs
- Forgetting next-hop-self
- RR in data path causing bottlenecks
- Missing redundancy

Module 9 Preview: Advanced Routing Policy Features

Now that we can deploy BGP at scale, we need sophisticated policies to control routing in complex scenarios. We'll explore advanced policy features that give you surgical control over route manipulation.

Module 9: Advanced Routing Policy Features

Beyond Basic Policies - The Precision Tools

Basic policies are like using a sledgehammer. Advanced policy features are like using a surgeon's scalpel - precise, powerful, and capable of complex operations.

Evolution of Policy Needs:

```
Simple: "Accept these networks"
    ↓
Advanced: "Accept these networks, but only if they come from
          specific ASes, have certain communities, and prefix
          length between /20-/24, then modify attributes based
          on time of day"
```

Advanced Route Filter Options

Basic route filters use "exact", "longer", and "orlonger". Advanced filters give you precise control:

1. Upto - Prefix Length Range

```
# Accept 10.0.0.0/8 and prefixes up to /24
from route-filter 10.0.0.0/8 upto /24

# This matches:
# 10.0.0.0/8  ✓
# 10.0.0.0/16 ✓
# 10.1.0.0/24 ✓
# 10.1.1.0/28 ✗ (too long)
```

2. Prefix-Length-Range - Specific Lengths

```
# Only accept /20 through /24 prefixes within 10.0.0.0/8
from route-filter 10.0.0.0/8 prefix-length-range /20-/24

# This matches:
# 10.0.0.0/8  ✗ (too short)
# 10.0.0.0/16 ✗ (too short)
# 10.1.0.0/20 ✓
# 10.1.0.0/24 ✓
# 10.1.1.0/28 ✗ (too long)
```

3. Through - Range of Prefixes

```
# Match range of prefixes
from route-filter 10.0.0.0/16 through 10.0.255.0/24

# Matches any prefix between these values
```

Prefix Lists - Reusable Route Filters

Prefix lists are like creating a phone book of networks you reference multiple times:

```
# Define the prefix list
set policy-options prefix-list CUSTOMER-PREFIXES 192.168.0.0/24
set policy-options prefix-list CUSTOMER-PREFIXES 192.168.1.0/24
set policy-options prefix-list CUSTOMER-PREFIXES 192.168.2.0/24
set policy-options prefix-list CUSTOMER-PREFIXES 172.16.0.0/20

# Use in multiple policies
set policy-options policy-statement POLICY1 term 1 from prefix-list CUSTOMER-PREFIXES
set policy-options policy-statement POLICY2 term 1 from prefix-list CUSTOMER-PREFIXES
```

Advanced Prefix List Usage:

```
# Prefix list with apply-path (dynamic!)
set policy-options prefix-list CONNECTED-ROUTES apply-path "interfaces <*> unit <*> family inet address <*>"
# Automatically includes all configured interface addresses!

# Hierarchical prefix lists
set policy-options prefix-list-filter CUSTOMERS orlonger
set policy-options policy-statement CHECK-CUSTOMERS from prefix-list-filter CUSTOMERS orlonger
```

Route Filters with Mixed Prefix Lengths

Real-world filtering often requires different length restrictions for different prefixes:

```
# Complex filter example
set policy-options policy-statement COMPLEX-FILTER term RFC1918 from route-filter 10.0.0.0/8 prefix-length-range /8-/24
set policy-options policy-statement COMPLEX-FILTER term RFC1918 from route-filter 172.16.0.0/12 prefix-length-range /12-/24
set policy-options policy-statement COMPLEX-FILTER term RFC1918 from route-filter 192.168.0.0/16 prefix-length-range
/16-/24
set policy-options policy-statement COMPLEX-FILTER term RFC1918 then reject
```

```
# Customer-specific lengths
set policy-options policy-statement CUSTOMER-LENGTHS term CUST-A from route-filter 203.0.113.0/24 prefix-length-range
/24-/28
set policy-options policy-statement CUSTOMER-LENGTHS term CUST-B from route-filter 198.51.100.0/24 prefix-length-range
/24-/32
```

Policy Subroutines - Modular Policies

Think of subroutines as functions you can call from other policies:

```
# Define reusable subroutine
set policy-options policy-statement SET-COMMUNITY-CUSTOMER then community add CUSTOMER
set policy-options policy-statement SET-COMMUNITY-CUSTOMER then next policy

set policy-options policy-statement SET-LOCAL-PREF-150 then local-preference 150
set policy-options policy-statement SET-LOCAL-PREF-150 then next policy

# Call from main policy
set policy-options policy-statement MAIN-POLICY term CUSTOMERS from protocol bgp
set policy-options policy-statement MAIN-POLICY term CUSTOMERS from neighbor 192.168.1.1
set policy-options policy-statement MAIN-POLICY term CUSTOMERS then policy SET-COMMUNITY-CUSTOMER
set policy-options policy-statement MAIN-POLICY term CUSTOMERS then policy SET-LOCAL-PREF-150
set policy-options policy-statement MAIN-POLICY term CUSTOMERS then accept
```

AS Path Regular Expressions

AS Path regex is like pattern matching for the Internet:

Basic Patterns:

```
# Define AS path patterns
set policy-options as-path SINGLE-AS "^65001$"           # Exactly AS 65001
set policy-options as-path VIA-AS701 ".* 701 .*"         # Through AS 701
set policy-options as-path CUSTOMER-AS "^65001 65002$"   # Customer with one AS hop
set policy-options as-path ANY-PRIVATE "^(64512-65535)+$" # Only private ASes
```

Complex Patterns:

```
# Regex operators:
# ^ = Start of path
# $ = End of path
# . = Any single AS
# * = Zero or more of previous
# + = One or more of previous
# ? = Zero or one of previous
# | = OR operator
# () = Grouping
# [] = Character class

# Advanced examples
set policy-options as-path LONG-PATH "^([0-9]+) ([0-9]+) ([0-9]+) ([0-9]+) .*"  # 4+ AS hops
set policy-options as-path DIRECT-OR-ONE-HOP "^([0-9]+)$|^([0-9]+) ([0-9]+)$"   # 0 or 1 hop
set policy-options as-path PREPENDED "^([0-9]+) \1+"                             # Prepended AS
```

Using AS Path in Policy:

```
set policy-options policy-statement AS-PATH-FILTER term PREFER-DIRECT from as-path SINGLE-AS
set policy-options policy-statement AS-PATH-FILTER term PREFER-DIRECT then local-preference 200

set policy-options policy-statement AS-PATH-FILTER term AVOID-LONG from as-path LONG-PATH
set policy-options policy-statement AS-PATH-FILTER term AVOID-LONG then local-preference 50
```

Community-Based Policy Control

Communities enable sophisticated policy signaling:

```
# Define community structure
set policy-options community BLACKHOLE members 65001:666
set policy-options community PREPEND-1 members 65001:101
set policy-options community PREPEND-2 members 65001:102
set policy-options community PREPEND-3 members 65001:103
set policy-options community NO-EXPORT-PEERS members 65001:200

# Act on communities
set policy-options policy-statement PROCESS-COMMUNITIES term BLACKHOLE from community BLACKHOLE
set policy-options policy-statement PROCESS-COMMUNITIES term BLACKHOLE then discard

set policy-options policy-statement PROCESS-COMMUNITIES term PREPEND-1X from community PREPEND-1
set policy-options policy-statement PROCESS-COMMUNITIES term PREPEND-1X then as-path-prepend "65001"

set policy-options policy-statement PROCESS-COMMUNITIES term PREPEND-2X from community PREPEND-2
set policy-options policy-statement PROCESS-COMMUNITIES term PREPEND-2X then as-path-prepend "65001 65001"
```

Advanced Policy Logic

1. Conditional Policy Application

```
# If-then-else logic
set policy-options policy-statement CONDITIONAL term IF-CUSTOMER from protocol bgp
set policy-options policy-statement CONDITIONAL term IF-CUSTOMER from community CUSTOMER
set policy-options policy-statement CONDITIONAL term IF-CUSTOMER then local-preference 150

set policy-options policy-statement CONDITIONAL term ELSE-IF-PEER from protocol bgp
set policy-options policy-statement CONDITIONAL term ELSE-IF-PEER from community PEER
set policy-options policy-statement CONDITIONAL term ELSE-IF-PEER then local-preference 100

set policy-options policy-statement CONDITIONAL term ELSE then local-preference 50
```

2. Policy Chains with Conditions

```
# Chain multiple policies based on conditions
set policy-options policy-statement ROUTER term FIRST from protocol bgp
set policy-options policy-statement ROUTER term FIRST then policy CUSTOMER-POLICY

set policy-options policy-statement ROUTER term SECOND from protocol ospf
set policy-options policy-statement ROUTER term SECOND then policy OSPF-POLICY

set policy-options policy-statement ROUTER term DEFAULT then policy DEFAULT-POLICY
```

Route Filter Evaluation Order

Understanding evaluation order is crucial:

```
# Evaluation order (most specific to least specific):
# 1. Longer prefix length
# 2. Route-filter type (exact > upto > orlonger > prefix-length-range)
# 3. Order in configuration

# Example demonstrating precedence
set policy-options policy-statement PRECEDENCE term 1 from route-filter 10.0.0.0/8 orlonger
set policy-options policy-statement PRECEDENCE term 1 then local-preference 100

set policy-options policy-statement PRECEDENCE term 2 from route-filter 10.1.0.0/16 exact
set policy-options policy-statement PRECEDENCE term 2 then local-preference 200

# 10.1.0.0/16 matches term 2 (more specific)
# 10.2.0.0/16 matches term 1
```

Complex Real-World Policy Example

Scenario: ISP Customer Categories with Traffic Engineering

```
# Community definitions
set policy-options community CUST-GOLD members 65001:100
set policy-options community CUST-SILVER members 65001:200
set policy-options community CUST-BRONZE members 65001:300
set policy-options community UPSTREAM-1 members 65001:1001
set policy-options community UPSTREAM-2 members 65001:1002
set policy-options community UPSTREAM-3 members 65001:1003

# Prefix lists
set policy-options prefix-list BOGONS 0.0.0.0/8
set policy-options prefix-list BOGONS 10.0.0.0/8
set policy-options prefix-list BOGONS 127.0.0.0/8
set policy-options prefix-list BOGONS 169.254.0.0/16
set policy-options prefix-list BOGONS 172.16.0.0/12
set policy-options prefix-list BOGONS 192.168.0.0/16
set policy-options prefix-list BOGONS 224.0.0.0/3

# AS path definitions
set policy-options as-path TRANSIT-FREE "^(174|209|701|1239|1299|2914|3257|3356|3491|5511|6453|6461|6762|7018)$"

# Inbound customer policy
set policy-options policy-statement FROM-CUSTOMERS term DENY-BOGONS from prefix-list BOGONS
set policy-options policy-statement FROM-CUSTOMERS term DENY-BOGONS then reject

set policy-options policy-statement FROM-CUSTOMERS term DENY-LONG-PREFIX from route-filter 0.0.0.0/0 prefix-length-range
/25-/32
set policy-options policy-statement FROM-CUSTOMERS term DENY-LONG-PREFIX then reject

set policy-options policy-statement FROM-CUSTOMERS term TAG-GOLD from neighbor 192.168.1.1
set policy-options policy-statement FROM-CUSTOMERS term TAG-GOLD then {
    community add CUST-GOLD;
    local-preference 150;
    next policy;
}

set policy-options policy-statement FROM-CUSTOMERS term TAG-SILVER from neighbor 192.168.2.1
set policy-options policy-statement FROM-CUSTOMERS term TAG-SILVER then {
    community add CUST-SILVER;
    local-preference 120;
    next policy;
}

set policy-options policy-statement FROM-CUSTOMERS term CHECK-PREFIX-LIMIT from protocol bgp
set policy-options policy-statement FROM-CUSTOMERS term CHECK-PREFIX-LIMIT from route-filter 0.0.0.0/0 prefix-length-range
/8-/24
set policy-options policy-statement FROM-CUSTOMERS term CHECK-PREFIX-LIMIT then accept

set policy-options policy-statement FROM-CUSTOMERS term DENY-REST then reject

# Outbound upstream policy with community control
set policy-options policy-statement TO-UPSTREAM term GOLD-CUSTOMERS from community CUST-GOLD
set policy-options policy-statement TO-UPSTREAM term GOLD-CUSTOMERS to neighbor 198.51.100.1
set policy-options policy-statement TO-UPSTREAM term GOLD-CUSTOMERS then {
    # Gold customers get no prepend
    next term;
}

set policy-options policy-statement TO-UPSTREAM term SILVER-CUSTOMERS from community CUST-SILVER
set policy-options policy-statement TO-UPSTREAM term SILVER-CUSTOMERS to neighbor 198.51.100.1
set policy-options policy-statement TO-UPSTREAM term SILVER-CUSTOMERS then {
    as-path-prepend "65001";
    next term;
}

set policy-options policy-statement TO-UPSTREAM term BRONZE-CUSTOMERS from community CUST-BRONZE
set policy-options policy-statement TO-UPSTREAM term BRONZE-CUSTOMERS to neighbor 198.51.100.1
set policy-options policy-statement TO-UPSTREAM term BRONZE-CUSTOMERS then {
    as-path-prepend "65001 65001";
    next term;
}
```

```
set policy-options policy-statement TO-UPSTREAM term ACCEPT-CUSTOMERS from community [ CUST-GOLD CUST-SILVER CUST-BRONZE ]
set policy-options policy-statement TO-UPSTREAM term ACCEPT-CUSTOMERS then accept

set policy-options policy-statement TO-UPSTREAM term DENY-REST then reject
```

Policy Testing and Debugging

1. Test Policy Without Applying

```
test policy FROM-CUSTOMERS 192.168.1.0/24
# Shows what would happen

show policy FROM-CUSTOMERS
# Shows policy logic flow
```

2. Policy Damping and Flap Statistics

```
# Configure damping policy
set policy-options damping AGGRESSIVE half-life 15
set policy-options damping AGGRESSIVE reuse 750
set policy-options damping AGGRESSIVE suppress 3000
set policy-options damping AGGRESSIVE max-suppress 60

set policy-options policy-statement DAMPING term APPLY-DAMPING from route-filter 0.0.0.0/0 upto /23
set policy-options policy-statement DAMPING term APPLY-DAMPING then damping AGGRESSIVE
```

3. Trace Policy Evaluation

```
# Enable policy tracing
set policy-options policy-statement MY-POLICY term 1 then trace

# View traces
show log policy-trace
```

Building Advanced Policy Instincts

Design Patterns:

1. **Layered Filtering**

   ```
   Layer 1: Security (bogons, prefix length)
   Layer 2: Classification (tag with communities)
   Layer 3: Preference (set local preference)
   Layer 4: Traffic Engineering (AS path prepend)
   ```

2. **Community-Driven Policies**
   - Tag on ingress
   - Act throughout network
   - Signal to upstreams
3. **Fail-Safe Defaults**
   - Always end with explicit reject
   - Use prefix limits
   - Validate prefix lengths

Common Advanced Mistakes:

1. **Regex Anchoring**

   ```
   # Bad: Matches AS 701 anywhere
   as-path TRANSIT "701"

   # Good: Matches only if directly connected
   as-path TRANSIT "^701$"
   ```

2. **Order Dependencies**

   ```
   # Terms evaluated in order
   # Put most specific first
   ```

3. **Policy Chain Loops**

   ```
   # Avoid circular policy references
   # Use "next term" carefully
   ```

Performance Considerations

1. Optimize Route Filters

```
# Use prefix-lists for static lists
# More efficient than multiple route-filters

# Good:
from prefix-list CUSTOMERS

# Less efficient:
from route-filter 192.168.1.0/24 exact
from route-filter 192.168.2.0/24 exact
from route-filter 192.168.3.0/24 exact
```

2. Minimize Regex Complexity

```
# Complex regex = slow evaluation
# Pre-filter when possible

term 1 from protocol bgp  # Pre-filter
term 1 from as-path COMPLEX-REGEX
```

Module 10 Preview: Routing Instances

Now we'll explore how to create multiple virtual routers within a single physical router using routing instances. This enables VPNs, traffic separation, and multi-tenant deployments.

Module 10: Routing Instances

What are Routing Instances? - Virtual Routers

Routing instances create multiple independent routing tables within a single physical router. Think of it as running multiple virtual routers inside one box.

Analogy:

```
Physical Router = Apartment Building
Routing Instances = Individual Apartments
- Each has its own address space
- Each has its own routing table
- Isolated from each other (by default)
- Can share common resources
```

Why Use Routing Instances?

Common Use Cases:

```
1. VPN Services (L3VPN)
2. Traffic Separation (Security Zones)
3. Multi-Tenant Environments
4. Management VRF
5. Internet in a VRF
6. Overlapping IP Address Spaces
```

Types of Routing Instances

1. Virtual Router (Most Common)

```
- Full routing functionality
- Separate routing protocols
- Used for VRFs/VPNs
```

2. VRF (VPN Routing and Forwarding)

```
- Specifically for MPLS L3VPNs
- Includes VPN-specific features
- Route distinguishers/targets
```

3. Virtual Switch

```
- Layer 2 instance
- For VPLS/L2VPN services
```

4. Forwarding

```
- Simple forwarding table
- No routing protocols
- Filter-based forwarding
```

Junos CLI Pattern #14: Basic Routing Instance

```
# Create a routing instance
set routing-instances CUSTOMER-A instance-type virtual-router

# Assign interfaces
set routing-instances CUSTOMER-A interface ge-0/0/1.0
set routing-instances CUSTOMER-A interface ge-0/0/2.0

# The pattern breakdown:
# - routing-instances: Container for all instances
# - CUSTOMER-A: Instance name (your choice)
# - instance-type: Defines capabilities
# - interface: Assigns physical/logical interfaces
```

Complete Virtual Router Configuration

```
# Step 1: Create the instance
set routing-instances CUSTOMER-A instance-type virtual-router

# Step 2: Assign interfaces
set interfaces ge-0/0/1 unit 0 family inet address 192.168.1.1/24
set routing-instances CUSTOMER-A interface ge-0/0/1.0

# Step 3: Configure routing within instance
set routing-instances CUSTOMER-A routing-options static route 0.0.0.0/0 next-hop 192.168.1.254
set routing-instances CUSTOMER-A protocols ospf area 0.0.0.0 interface ge-0/0/1.0
```

The Default Instance - Understanding "Master"

The main routing table you've been using is actually a routing instance called "master":

```
Regular command:                    Full path:
show route                         show route table inet.0
set protocols ospf...              set routing-instances master protocols ospf...

All commands without "routing-instances" apply to master
```

Working with Routing Instances

Entering Instance Context:

```
# Method 1: Full path
show route table CUSTOMER-A.inet.0

# Method 2: Set context
set cli logical-system CUSTOMER-A
show route  # Now shows CUSTOMER-A routes

# Return to master
clear cli logical-system
```

Routing Table Naming:

```
Instance: master
Tables: inet.0, inet.1, inet.2, inet.3

Instance: CUSTOMER-A
Tables: CUSTOMER-A.inet.0, CUSTOMER-A.inet.1, etc.
```

Inter-Instance Route Sharing

By default, instances are isolated. You can share routes between them:

Method 1: RIB Groups (Routing Information Base Groups)

```
# Create RIB group
set routing-options rib-groups SHARE-CONNECTED import-rib [ inet.0 CUSTOMER-A.inet.0 CUSTOMER-B.inet.0 ]

# Apply to protocol
set routing-options interface-routes rib-group inet SHARE-CONNECTED

# This shares directly connected routes to all specified tables
```

Method 2: Instance Import/Export

```
# Export from instance
set routing-instances CUSTOMER-A routing-options instance-export EXPORT-CONNECTED
set policy-options policy-statement EXPORT-CONNECTED term 1 from protocol direct
set policy-options policy-statement EXPORT-CONNECTED term 1 then accept

# Import to instance
set routing-instances CUSTOMER-B routing-options instance-import FROM-CUSTOMER-A
set policy-options policy-statement FROM-CUSTOMER-A term 1 from instance CUSTOMER-A
set policy-options policy-statement FROM-CUSTOMER-A term 1 then accept
```

Method 3: Logical Tunnel Interfaces

```
# Create logical tunnel
set interfaces lt-0/0/0 unit 1 encapsulation ethernet
set interfaces lt-0/0/0 unit 1 peer-unit 2
set interfaces lt-0/0/0 unit 1 family inet address 10.0.0.1/30

set interfaces lt-0/0/0 unit 2 encapsulation ethernet
set interfaces lt-0/0/0 unit 2 peer-unit 1
set interfaces lt-0/0/0 unit 2 family inet address 10.0.0.2/30

# Assign to different instances
set routing-instances CUSTOMER-A interface lt-0/0/0.1
set routing-instances CUSTOMER-B interface lt-0/0/0.2

# Now run routing protocol between instances!
```

VRF Configuration for MPLS L3VPN

```
# VRF instance configuration
set routing-instances CUSTOMER-VPN instance-type vrf
set routing-instances CUSTOMER-VPN interface ge-0/0/1.0
set routing-instances CUSTOMER-VPN route-distinguisher 65001:100
set routing-instances CUSTOMER-VPN vrf-target target:65001:100

# VRF-specific features:
# - route-distinguisher: Makes routes globally unique
# - vrf-target: Controls import/export in MPLS network
```

Advanced Routing Instance Features

1. Internet in a VRF

```
# Put Internet routing in VRF
set routing-instances INTERNET instance-type virtual-router
set routing-instances INTERNET interface ge-0/0/0.0  # ISP connection

# Configure BGP within instance
set routing-instances INTERNET protocols bgp group ISP type external
set routing-instances INTERNET protocols bgp group ISP peer-as 701
set routing-instances INTERNET protocols bgp group ISP neighbor 198.51.100.1
```

2. Management VRF

```
# Separate management traffic
set routing-instances MGMT instance-type virtual-router
set routing-instances MGMT interface ge-0/0/10.0
set routing-instances MGMT routing-options static route 0.0.0.0/0 next-hop 10.0.0.1

# System services bind to instance
set system services ssh routing-instance MGMT
set system services netconf ssh routing-instance MGMT
```

3. Route Leaking Between VRFs

```
# Selective route sharing
set routing-instances VRF-A routing-options instance-import IMPORT-FROM-VRF-B
set routing-instances VRF-A routing-options instance-export EXPORT-TO-VRF-B

set policy-options policy-statement IMPORT-FROM-VRF-B term SERVER-NET from instance VRF-B
set policy-options policy-statement IMPORT-FROM-VRF-B term SERVER-NET from route-filter 10.100.0.0/24 exact
set policy-options policy-statement IMPORT-FROM-VRF-B term SERVER-NET then accept
set policy-options policy-statement IMPORT-FROM-VRF-B term DENY then reject
```

Routing Instance Design Patterns

Pattern 1: Hub and Spoke VRF

```
     [Shared Services VRF]
          /    |    \
        /      |      \
   [VRF-A]  [VRF-B]  [VRF-C]

Spoke VRFs import specific routes from hub
Hub imports all routes from spokes
```

Configuration:

```
# Hub VRF
set routing-instances SHARED-SERVICES routing-options instance-import FROM-ALL-SPOKES
set policy-options policy-statement FROM-ALL-SPOKES term 1 from instance [ VRF-A VRF-B VRF-C ]
set policy-options policy-statement FROM-ALL-SPOKES term 1 then accept

# Spoke VRF
```

```
set routing-instances VRF-A routing-options instance-import FROM-SHARED
set policy-options policy-statement FROM-SHARED term 1 from instance SHARED-SERVICES
set policy-options policy-statement FROM-SHARED term 1 from route-filter 10.100.0.0/16 orlonger
set policy-options policy-statement FROM-SHARED term 1 then accept
```

Pattern 2: Extranet Services

```
# Customer A needs access to Customer B's web server
set policy-options policy-statement A-TO-B-EXTRANET term WEB-SERVER from instance CUSTOMER-B
set policy-options policy-statement A-TO-B-EXTRANET term WEB-SERVER from route-filter 192.168.100.10/32 exact
set policy-options policy-statement A-TO-B-EXTRANET term WEB-SERVER then accept

set routing-instances CUSTOMER-A routing-options instance-import A-TO-B-EXTRANET
```

Troubleshooting Pattern #6: Routing Instance Issues

Problem 1: Routes Not Appearing in VRF

```
# Check interface assignment
show interfaces routing-instance CUSTOMER-A

# Verify route table
show route table CUSTOMER-A.inet.0

# Check if interface is in correct instance
show configuration routing-instances CUSTOMER-A interface
```

Problem 2: Cannot Ping Across Instances

```
# Ping from specific instance
ping 192.168.1.1 routing-instance CUSTOMER-A

# Traceroute from instance
traceroute 192.168.1.1 routing-instance CUSTOMER-A

# Check route in source instance
show route 192.168.1.1 table CUSTOMER-A.inet.0
```

Problem 3: Route Leaking Not Working

```
# Verify RIB group configuration
show configuration routing-options rib-groups

# Check policy evaluation
test policy IMPORT-POLICY 192.168.1.0/24

# Verify instance-import applied
show configuration routing-instances CUSTOMER-A routing-options instance-import
```

Complex Multi-Tenant Example

Scenario: Service Provider with Multiple Customers

```
# Customer A Configuration
set interfaces ge-0/0/1 unit 100 vlan-id 100
set interfaces ge-0/0/1 unit 100 family inet address 10.1.1.1/30
set routing-instances CUSTOMER-A instance-type virtual-router
set routing-instances CUSTOMER-A interface ge-0/0/1.100
set routing-instances CUSTOMER-A protocols bgp group CE type external
set routing-instances CUSTOMER-A protocols bgp group CE peer-as 65100
set routing-instances CUSTOMER-A protocols bgp group CE neighbor 10.1.1.2

# Customer B Configuration
set interfaces ge-0/0/1 unit 200 vlan-id 200
set interfaces ge-0/0/1 unit 200 family inet address 10.2.1.1/30
set routing-instances CUSTOMER-B instance-type virtual-router
set routing-instances CUSTOMER-B interface ge-0/0/1.200
set routing-instances CUSTOMER-B protocols bgp group CE type external
set routing-instances CUSTOMER-B protocols bgp group CE peer-as 65200
set routing-instances CUSTOMER-B protocols bgp group CE neighbor 10.2.1.2

# Shared Internet Service
set routing-instances INTERNET instance-type virtual-router
set routing-instances INTERNET interface ge-0/0/0.0
set routing-instances INTERNET protocols bgp group ISP type external
set routing-instances INTERNET protocols bgp group ISP peer-as 701
set routing-instances INTERNET protocols bgp group ISP neighbor 198.51.100.1

# Allow customers to access Internet
set routing-options rib-groups INET-ACCESS import-rib [ INTERNET.inet.0 CUSTOMER-A.inet.0 CUSTOMER-B.inet.0 ]
set routing-instances INTERNET routing-options interface-routes rib-group inet INET-ACCESS
```

Performance and Scaling Considerations

1. Resource Usage

```
Each routing instance consumes:
- Memory for routing tables
- CPU for protocol processing
- RIB/FIB entries

Plan capacity accordingly
```

2. Best Practices

```
# Limit routing table size
set routing-instances CUSTOMER-A routing-options maximum-routes 10000 threshold 90

# Use aggregate labels for MPLS
set routing-instances CUSTOMER-A routing-options aggregate-label community 65001:100
```

Building Routing Instance Instincts

Design Principles:

1. Start simple - virtual-router for most cases
2. Use consistent naming conventions
3. Document instance purposes
4. Plan IP addressing to avoid conflicts
5. Consider route table growth

Common Pitfalls:

1. Forgetting to assign interfaces
2. Wrong instance type for use case
3. Route leaking loops
4. Overlapping IP addresses
5. Missing instance in commands

Quick Checks:

```
# What instances exist?
show routing-instances

# What's in each instance?
show route summary table CUSTOMER-A.inet.0

# Interface assignments?
show interfaces routing-instance
```

Real-World Deployment Example

Enterprise with DMZ and Guest Networks:

```
# Production Network (Default)
set interfaces ge-0/0/0 unit 0 family inet address 10.0.0.1/24

# DMZ Network
set routing-instances DMZ instance-type virtual-router
set routing-instances DMZ interface ge-0/0/1.0
set routing-instances DMZ routing-options static route 0.0.0.0/0 next-hop 192.168.1.1

# Guest Network
set routing-instances GUEST instance-type virtual-router
set routing-instances GUEST interface ge-0/0/2.0
set routing-instances GUEST protocols ospf area 0.0.0.0 interface ge-0/0/2.0

# Firewall filter between instances
set firewall family inet filter DMZ-TO-PROD term ALLOW-HTTP from source-prefix-list DMZ-SERVERS
set firewall family inet filter DMZ-TO-PROD term ALLOW-HTTP from protocol tcp
set firewall family inet filter DMZ-TO-PROD term ALLOW-HTTP from destination-port 80
set firewall family inet filter DMZ-TO-PROD term ALLOW-HTTP then accept
set firewall family inet filter DMZ-TO-PROD term DENY-ALL then discard

# Apply filter
set interfaces lt-0/0/0 unit 1 family inet filter input DMZ-TO-PROD
```

Module 11 Preview: Load Balancing

Now that we can separate traffic into instances, let's explore how to distribute traffic across multiple paths efficiently using various load balancing techniques.

Module 11: Load Balancing

Understanding Load Balancing - Traffic Distribution

Load balancing in routing is about utilizing multiple paths efficiently. Instead of having expensive backup links sitting idle, we distribute traffic across all available paths.

The Evolution of Path Usage:

```
Traditional:                   Load Balanced:
Primary Path: ████████████     Path 1: ██████
Backup Path:  ____________     Path 2: ██████
(Backup unused until failure)  (Both paths actively used)
```

Types of Load Balancing

1. Per-Packet (Round-Robin)

```
Packet 1 → Path A
Packet 2 → Path B
Packet 3 → Path A
Packet 4 → Path B

Pros: Perfect distribution
Cons: Packet reordering, issues with stateful devices
```

2. Per-Flow (Hash-Based)

```
Flow 1 (Src:10.1.1.1 Dst:20.1.1.1) → Path A (always)
Flow 2 (Src:10.1.1.2 Dst:20.1.1.1) → Path B (always)
```

```
Pros: No reordering, preserves flow state
Cons: May have uneven distribution
```

ECMP (Equal-Cost Multi-Path)

ECMP is the foundation of load balancing - when multiple paths have the same metric, use them all:

```
          [R2]
         /    \
    10  /      \ 10
       /        \
   [R1]          [R4]
       \        /
    10  \      / 10
         \    /
          [R3]

Two equal-cost paths: R1→R2→R4 and R1→R3→R4
```

Junos CLI Pattern #15: Basic Load Balancing

```
# Enable load balancing (per-flow is default)
set policy-options policy-statement LOAD-BALANCE then load-balance per-packet

# Apply to forwarding table
set routing-options forwarding-table export LOAD-BALANCE

# The pattern breakdown:
# - Create policy to enable load balancing
# - Apply to forwarding-table export
# - Affects how traffic is actually forwarded
```

Complete Load Balancing Configuration

Per-Flow Load Balancing (Default/Recommended):

```
# This is default behavior - no configuration needed
# But you can explicitly configure:
set routing-options forwarding-table hash-key family inet layer-3
set routing-options forwarding-table hash-key family inet layer-4

# What gets hashed:
# - Source IP
# - Destination IP
# - Source Port (Layer 4)
# - Destination Port (Layer 4)
# - Protocol
```

Per-Packet Load Balancing:

```
# Define policy
set policy-options policy-statement PER-PACKET-LB then load-balance per-packet

# Apply globally
set routing-options forwarding-table export PER-PACKET-LB

# Or apply selectively
set policy-options policy-statement SELECTIVE-LB from route-filter 10.0.0.0/8 orlonger
set policy-options policy-statement SELECTIVE-LB then load-balance per-packet
set routing-options forwarding-table export SELECTIVE-LB
```

Load Balancing with Different Protocols

1. Static Routes with Multiple Next-Hops:

```
# Single command creates ECMP
set routing-options static route 192.168.1.0/24 next-hop [ 10.1.1.1 10.1.2.1 ]

# Or use multiple statements
set routing-options static route 192.168.1.0/24 next-hop 10.1.1.1
set routing-options static route 192.168.1.0/24 next-hop 10.1.2.1

# Verify ECMP
show route 192.168.1.0/24
# Shows multiple next-hops with "Multipath" tag
```

2. OSPF Load Balancing:

```
# OSPF automatically does ECMP for equal-cost paths
# Ensure interfaces have same metric:
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 metric 10
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0 metric 10

# Control maximum paths
set protocols ospf maximum-paths 4
```

3. BGP Load Balancing:

```
# BGP requires special configuration for load balancing
# Option 1: Multipath
set protocols bgp group INTERNAL multipath

# Option 2: Add-path (receive multiple paths)
set protocols bgp group INTERNAL family inet unicast add-path receive

# Option 3: Multiple sessions to same peer
set protocols bgp group ISP neighbor 198.51.100.1 multihop
set protocols bgp group ISP neighbor 198.51.100.2 multihop
```

Advanced Load Balancing Features

1. Weighted Load Balancing:

```
# Using bandwidth for weight
set interfaces ge-0/0/0 unit 0 bandwidth 1g
set interfaces ge-0/0/1 unit 0 bandwidth 10g

# Create policy to use bandwidth
set policy-options policy-statement WEIGHTED-LB then load-balance bandwidth
```

2. Adaptive Load Balancing:

```
# Monitor utilization and adjust
set routing-options forwarding-table adaptive-load-balance
set routing-options forwarding-table adaptive-load-balance tolerance 20

# Rebalances flows when utilization differs by >20%
```

3. Consistent Hashing:

```
# Minimize flow disruption during changes
set routing-options forwarding-table consistent-load-balance
```

Load Balancing Across LAGs (Link Aggregation)

```
# Configure LAG
set interfaces ae0 aggregated-ether-options minimum-links 2
set interfaces ae0 aggregated-ether-options link-speed 10g
set interfaces ae0 aggregated-ether-options lacp active

# Add member interfaces
```

```
set interfaces ge-0/0/0 ether-options 802.3ad ae0
set interfaces ge-0/0/1 ether-options 802.3ad ae0

# Configure load balancing on LAG
set interfaces ae0 aggregated-ether-options load-balance adaptive
set interfaces ae0 aggregated-ether-options load-balance adaptive scan-interval 5
```

Monitoring Load Balancing

Key Commands:

```
# View load distribution
show interfaces ae0 extensive | match "Distribution"

# See forwarding table with next-hops
show route forwarding-table destination 192.168.1.0/24

# Monitor interface statistics
monitor interface ge-0/0/0
monitor interface ge-0/0/1

# Check hash computation
show route forwarding-table destination 192.168.1.1 source 10.1.1.1
```

Understanding Output:

```
user@router> show route 10.0.0.0/24

10.0.0.0/24   *[OSPF/10] 00:05:00, metric 20
              > to 192.168.1.1 via ge-0/0/0.0
                to 192.168.2.1 via ge-0/0/1.0
                                    ^
                           Multiple next-hops = Load balancing
```

Troubleshooting Pattern #7: Load Balancing Issues

Problem 1: Traffic Using Only One Path

```
# Check 1: Are there multiple paths?
show route 192.168.1.0/24 detail

# Check 2: Is load balancing enabled?
show configuration routing-options forwarding-table

# Check 3: Hash distribution
show pfe statistics traffic hash

# Fix: Ensure proper hash keys
set routing-options forwarding-table hash-key family inet layer-4
```

Problem 2: Uneven Load Distribution

```
# Verify flow distribution
show interfaces statistics detail | match "bps|pps"

# Common cause: Few flows, same hash
# Solution 1: Use more hash fields
set routing-options forwarding-table hash-key family inet incoming-interface

# Solution 2: Per-packet for lab/non-production
set policy-options policy-statement PER-PACKET then load-balance per-packet
```

Problem 3: Application Issues with Load Balancing

```
# Symptoms: Out-of-order packets, TCP resets
# Check if using per-packet
show configuration policy-options policy-statement

# Switch to per-flow (default)
delete routing-options forwarding-table export PER-PACKET-LB
```

Load Balancing Design Patterns

Pattern 1: Data Center Load Balancing

```
# Multiple uplinks to spine switches
set routing-options static route 0.0.0.0/0 next-hop [ 10.0.0.1 10.0.0.2 10.0.0.3 10.0.0.4 ]

# ECMP with consistent hashing
set routing-options forwarding-table consistent-load-balance

# Maximum paths for scalability
set routing-options forwarding-table maximum-ecmp 64
```

Pattern 2: WAN Load Balancing

```
# Different bandwidth links
set interfaces ge-0/0/0 unit 0 bandwidth 100m
set interfaces ge-0/0/1 unit 0 bandwidth 1g

# Weighted load balancing
set policy-options policy-statement WEIGHTED-LB then load-balance bandwidth
set routing-options forwarding-table export WEIGHTED-LB
```

Pattern 3: Internet Load Balancing (Dual ISP)

```
# BGP multipath for same AS
set protocols bgp group ISP-PEERS multipath multiple-as

# Policy to enable load balancing
set policy-options policy-statement LB-INTERNET from route-filter 0.0.0.0/0 exact
set policy-options policy-statement LB-INTERNET then load-balance per-packet
```

Advanced Considerations

1. Polarization (Hash Bias)

```
Problem: Same hash result at each hop
[R1]→[R2]→[R3]→[R4]
All use upper path due to same hash

Solution: Vary hash algorithm
set routing-options forwarding-table hash-seed 12345
```

2. Asymmetric Routing

```
Forward: Client → Path A → Server
Return:  Server → Path B → Client

Can cause:
- Stateful firewall issues
- TCP optimization problems
```

3. Load Balancing with Stateful Services

```
# Firewall filter to ensure symmetry
set firewall filter SYMMETRIC term 1 from source-address 10.1.1.0/24
set firewall filter SYMMETRIC term 1 then routing-instance PATH-A
set firewall filter SYMMETRIC term 2 from source-address 10.1.2.0/24
set firewall filter SYMMETRIC term 2 then routing-instance PATH-B
```

Real-World Load Balancing Scenario

Multi-Homed Branch Office:

```
# Two WAN links - MPLS and Internet
set interfaces ge-0/0/0 unit 0 description "MPLS 100Mbps"
set interfaces ge-0/0/0 unit 0 family inet address 172.16.1.2/30

set interfaces ge-0/0/1 unit 0 description "Internet 200Mbps"
set interfaces ge-0/0/1 unit 0 family inet address 192.168.1.2/30

# Static routes with different metrics
set routing-options static route 10.0.0.0/8 next-hop 172.16.1.1 metric 10
set routing-options static route 10.0.0.0/8 next-hop 192.168.1.1 metric 10

# Policy-based load balancing
set policy-options policy-statement CRITICAL-TRAFFIC from source-address 10.1.1.0/24
set policy-options policy-statement CRITICAL-TRAFFIC then accept

set policy-options policy-statement USE-MPLS from policy CRITICAL-TRAFFIC
set policy-options policy-statement USE-MPLS then next-hop 172.16.1.1

set routing-options forwarding-table export USE-MPLS

# BFD for fast failover
set protocols bfd traceoptions file bfd-log
set protocols bfd traceoptions flag all

set routing-options static route 10.0.0.0/8 next-hop 172.16.1.1 bfd-liveness-detection minimum-interval 300
set routing-options static route 10.0.0.0/8 next-hop 192.168.1.1 bfd-liveness-detection minimum-interval 300
```

Performance Optimization

```
# Optimize hash computation
set routing-options forwarding-table hash-key family inet precompute

# Increase ECMP paths
set routing-options forwarding-table maximum-ecmp 32

# Fine-tune adaptive behavior
set routing-options forwarding-table adaptive-load-balance
set routing-options forwarding-table adaptive-load-balance scan-interval 30
set routing-options forwarding-table adaptive-load-balance tolerance 15
```

Building Load Balancing Instincts

Key Principles:

1. Per-flow is almost always right
2. Monitor actual distribution
3. Consider application requirements
4. Plan for asymmetry
5. Test failover scenarios

Quick Diagnostics:

```
# Is it working?
show route forwarding-table | match "ulst\|list"

# How's distribution?
show interfaces | match "bps|pps" | except "0 bps"

# Any errors?
show interfaces extensive | match "error|drop"
```

Common Mistakes:

1. Per-packet in production (causes reordering)
2. Not monitoring actual distribution
3. Forgetting about return path
4. Missing BFD for fast failover
5. Unequal metrics preventing ECMP

Day 3 Preview: High Availability Features

Tomorrow we'll explore VRRP for gateway redundancy, graceful restart for hitless upgrades, and advanced HA features like GRES and NSR. These build on load balancing to create truly resilient networks.

Module 12: VRRP (Virtual Router Redundancy Protocol)

Welcome to Day 3 - High Availability Focus

Today we shift focus from traffic distribution to high availability. While load balancing uses all paths, HA ensures service continues when components fail.

What is VRRP? - The Gateway Protection

VRRP solves a fundamental problem: hosts typically have one default gateway. If that gateway fails, they're isolated.

The Problem:

```
Hosts configured with:
Default Gateway: 192.168.1.1

If 192.168.1.1 fails → No connectivity!

Traditional solution: Manually change gateway
Better solution: VRRP automatic failover
```

How VRRP Works:

```
Physical Reality:             What Hosts See:
[Router-A: .2] [Router-B: .3]  [Virtual Router: .1]
      |              |                   |
      +------+-------+                   |
             |                           |
         [Switch]                    [Switch]
             |                           |
         [Hosts]                      [Hosts]
```

VRRP Concepts

Key Components:

1. **Virtual IP (VIP)** - The IP hosts use as gateway
2. **Virtual MAC** - 00:00:5E:00:01:XX (XX = VRID)
3. **Master Router** - Currently owns VIP
4. **Backup Router(s)** - Ready to take over
5. **VRID** - Virtual Router ID (1-255)
6. **Priority** - Determines master (higher wins)

VRRP States:

```
Initialize → Backup → Master
     ↑         ↓        ↓
```

```
      +----←-----+←--------+
          (on failure)
```

Junos CLI Pattern #16: Basic VRRP Configuration

```
# On Router A (Primary)
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 virtual-address 192.168.1.1
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 priority 200

# On Router B (Backup)
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.3/24 vrrp-group 1 virtual-address 192.168.1.1
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.3/24 vrrp-group 1 priority 100

# Pattern breakdown:
# - Configure under interface family inet address
# - vrrp-group NUMBER (must match on all routers)
# - virtual-address (the VIP)
# - priority (higher becomes master)
```

VRRP Priority and Preemption

Priority Rules:

- Range: 1-255
- Default: 100
- Higher priority = Master
- Priority 255 = Always master (owns VIP)
- Priority 0 = Release master role

Preemption (Default Enabled):

```
# If higher priority router comes online, it takes over

# Disable preemption
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 no-preempt

# Preempt delay (wait before taking over)
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 preempt-delay 60
```

Advanced VRRP Features

1. Authentication:

```
# Simple authentication
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 authentication-type simple
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 authentication-key SecretKey

# MD5 authentication
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 authentication-type md5
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 authentication-key "$9$encrypted"
```

2. Track Interface/Routes:

```
# Reduce priority if tracked interface fails
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 track interface ge-0/0/1.0 priority-cost 50

# If ge-0/0/1.0 fails: Priority = 200 - 50 = 150

# Track route existence
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 track route 0.0.0.0/0 routing-instance
default priority-cost 100
```

3. Multiple VRRP Groups:

```
# Load sharing with multiple groups
# Router A
```

```
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 virtual-address 192.168.1.1
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 priority 200

set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 2 virtual-address 192.168.1.254
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 2 priority 100

# Router B (opposite priorities)
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.3/24 vrrp-group 1 virtual-address 192.168.1.1
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.3/24 vrrp-group 1 priority 100

set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.3/24 vrrp-group 2 virtual-address 192.168.1.254
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.3/24 vrrp-group 2 priority 200

# Result: A is master for .1, B is master for .254
```

VRRP Timers

Default Timers:

- Advertisement Interval: 1 second
- Master Down Interval: 3 * Advertisement + Skew

Tuning Timers:

```
# Faster failover
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 advertise-interval 1
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 fast-interval 200

# fast-interval in milliseconds (200ms = 0.2 seconds)
```

VRRP Version Differences

VRRPv2 vs VRRPv3:

```
# VRRPv2 (default) - IPv4 only
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 version-2

# VRRPv3 - IPv4 and IPv6
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 version-3

# IPv6 VRRP
set interfaces ge-0/0/0 unit 0 family inet6 address 2001:db8::2/64 vrrp-inet6-group 1 virtual-inet6-address 2001:db8::1
set interfaces ge-0/0/0 unit 0 family inet6 address 2001:db8::2/64 vrrp-inet6-group 1 priority 200
```

Monitoring VRRP

```
# Show VRRP status
show vrrp
show vrrp detail
show vrrp interface ge-0/0/0.0

# Monitor state changes
monitor start messages | match VRRP

# Track statistics
show vrrp statistics
```

Reading VRRP Output:

```
Physical interface: ge-0/0/0.0, Unit: 0, Vlan-id: 0
  Interface state: up, Group: 1, State: master
  Priority: 200, Advertisement interval: 1, Authentication type: none
  Advertisement timer: 0.425s, Master router: 192.168.1.2
  Virtual router: 192.168.1.1, VIP count: 1, VIP: 192.168.1.1
  Virtual MAC: 00:00:5e:00:01:01
```

Troubleshooting Pattern #8: VRRP Issues

Problem 1: Both Routers Claim Master

```
# Symptom: Duplicate IP warnings, packet loss

# Check 1: VRID match?
show vrrp detail | match "Group:"

# Check 2: L2 connectivity?
monitor traffic interface ge-0/0/0.0 matching vrrp

# Common cause: Switch/VLAN misconfiguration
# Fix: Ensure both routers in same broadcast domain
```

Problem 2: Backup Not Taking Over

```
# Check 1: Can backup see master advertisements?
monitor traffic interface ge-0/0/0.0 no-resolve matching vrrp

# Check 2: Authentication mismatch?
show configuration interfaces ge-0/0/0.0 | match authentication

# Check 3: Preemption disabled?
show configuration interfaces ge-0/0/0.0 | match preempt
```

Problem 3: Flapping Between Master/Backup

```
# Check interface stability
show interfaces ge-0/0/0 extensive | match flap

# Increase advertisement interval
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 advertise-interval 2

# Add preempt delay
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 preempt-delay 180
```

VRRP Design Patterns

Pattern 1: Active/Standby with Tracking

```
# Track upstream connectivity
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 priority 200
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 track interface ge-0/0/1.0 priority-cost 150

# If upstream fails, priority becomes 50, forcing failover
```

Pattern 2: Load Balanced VRRP

```
# Multiple groups, alternating masters
# Configure DHCP to distribute groups:
# 50% of clients get 192.168.1.1 (Group 1)
# 50% of clients get 192.168.1.254 (Group 2)
```

Pattern 3: VRRP with Routing Protocols

```
# Ensure OSPF/BGP follows VRRP state
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 passive

# Or use different metrics
set policy-options policy-statement VRRP-METRIC term MASTER from protocol direct
set policy-options policy-statement VRRP-METRIC term MASTER from interface ge-0/0/0.0
set policy-options policy-statement VRRP-METRIC term MASTER then metric 10

set policy-options policy-statement VRRP-METRIC term BACKUP from protocol direct
set policy-options policy-statement VRRP-METRIC term BACKUP then metric 100
```

Complex VRRP Scenario

Multi-Site VRRP with Stretched L2:

```
# Site A - Router 1 (Primary for VLAN 10)
set interfaces ge-0/0/0 unit 10 vlan-id 10
set interfaces ge-0/0/0 unit 10 family inet address 10.1.10.2/24 vrrp-group 10 virtual-address 10.1.10.1
set interfaces ge-0/0/0 unit 10 family inet address 10.1.10.2/24 vrrp-group 10 priority 200
set interfaces ge-0/0/0 unit 10 family inet address 10.1.10.2/24 vrrp-group 10 track interface ge-0/0/1.0 priority-cost 150

# Site A - Router 2 (Backup for VLAN 10)
set interfaces ge-0/0/0 unit 10 vlan-id 10
set interfaces ge-0/0/0 unit 10 family inet address 10.1.10.3/24 vrrp-group 10 virtual-address 10.1.10.1
set interfaces ge-0/0/0 unit 10 family inet address 10.1.10.3/24 vrrp-group 10 priority 150

# Site B - Router 3 (Tertiary for VLAN 10)
set interfaces ge-0/0/0 unit 10 vlan-id 10
set interfaces ge-0/0/0 unit 10 family inet address 10.1.10.4/24 vrrp-group 10 virtual-address 10.1.10.1
set interfaces ge-0/0/0 unit 10 family inet address 10.1.10.4/24 vrrp-group 10 priority 100
set interfaces ge-0/0/0 unit 10 family inet address 10.1.10.4/24 vrrp-group 10 preempt-delay 300

# Add accept-data for hairpinning
set interfaces ge-0/0/0 unit 10 family inet address 10.1.10.2/24 vrrp-group 10 accept-data
```

VRRP Best Practices

1. **Always Use Authentication**
   - Prevents rogue routers
   - Use MD5 over simple
2. **Track Critical Resources**
   - Upstream interfaces
   - Default routes
   - Critical services
3. **Document Priority Scheme**

   ```
   Priority Allocation:
   200-255: Primary routers
   150-199: Secondary routers
   100-149: Tertiary routers
   50-99:   Degraded state
   ```

4. **Plan IP Addressing**

   ```
   .1:   VRRP VIP (Group 1)
   .2:   Router A physical
   .3:   Router B physical
   .254: VRRP VIP (Group 2)
   ```

Integration with Other Features

VRRP + BFD:

```
# Faster failure detection
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 track bfd-liveness-detection neighbor
192.168.1.3
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 track bfd-liveness-detection minimum-
interval 100
```

VRRP + Routing Instances:

```
# VRRP per routing instance
set routing-instances CUSTOMER-A interface ge-0/0/1.100
set interfaces ge-0/0/1 unit 100 family inet address 172.16.1.2/24 vrrp-group 1 virtual-address 172.16.1.1
```

Building VRRP Instincts

Quick Status Check:

```
# Who's master?
show vrrp brief | match master

# Any state changes?
show log messages | match VRRP | last 20

# Track history
show vrrp track detail
```

Common Mistakes:

1. Mismatched VRID
2. Wrong VLAN/broadcast domain
3. Forgetting accept-data for services
4. Not tracking upstream connectivity
5. Authentication type mismatches

Design Decisions:

- Active/Standby vs Load Balanced?
- How many backup routers?
- What to track?
- Preemption strategy?
- Timer optimization?

Module 13 Preview: Graceful Restart and BFD

Next, we'll explore protocols that detect failures in milliseconds (BFD) and allow routers to restart without dropping traffic (Graceful Restart). These complement VRRP for complete high availability.

Module 14: GRES, NSR, and Unified ISSU

Hardware-Level High Availability

We've covered protocol redundancy (VRRP), fast detection (BFD), and graceful protocol restart. Now we tackle hardware redundancy - keeping services running even when hardware components fail or need upgrades.

The High Availability Stack:

```
Application Level:    VRRP (Gateway redundancy)
Protocol Level:       Graceful Restart, BFD
Hardware Level:       GRES, NSR, ISSU ← We are here
```

Understanding Juniper Hardware Architecture

Before diving into features, understand the architecture:

```
Juniper Router/Switch Architecture:

[Routing Engine 0 (Master)]     [Routing Engine 1 (Backup)]
- Control Plane                 - Standby
- Management                    - Ready to take over
- Routing Protocols
        ↓                               ↓
        =====================================
                    [Packet Forwarding Engine]
                    - Data Plane
```

```
                      - Hardware Forwarding
                      - Line Cards
```

GRES - Graceful Routing Engine Switchover

GRES allows the backup Routing Engine (RE) to take over from the master RE without interrupting packet forwarding.

Without GRES:

```
Master RE fails → No control plane → Protocols down → Forwarding stops
                                                            ↓
Backup RE boots → Protocols start → Convergence → 2-5 minutes downtime
```

With GRES:

```
Master RE fails → Backup RE takes over → Forwarding continues
                          ↓
                   <1 second switchover
```

How GRES Works

The GRES Process:

1. **Synchronization Phase**
   - Master RE continuously syncs to backup
   - Kernel state replicated
   - Interface states maintained
2. **Failure Detection**
   - Hardware monitoring
   - Heartbeat between REs
   - Automatic or manual switchover
3. **Switchover**
   - Backup becomes master
   - Preserves kernel state
   - PFE continues forwarding

Junos CLI Pattern #19: GRES Configuration

```
# Enable GRES (on both REs)
set chassis redundancy graceful-switchover

# That's it! But verify prerequisites:
# - Dual REs installed
# - Same Junos version
# - Commit synchronize enabled

# Enable configuration synchronization
set system commit synchronize

# Pattern breakdown:
# - chassis redundancy: Hardware HA features
# - graceful-switchover: Enable GRES
# - commit synchronize: Keep configs in sync
```

GRES Verification and Testing

```
# Verify GRES status
show system switchover

# Output:
Graceful switchover: On
Configuration database: Ready
Kernel database: Ready
Peer state: Steady State
```

```
# Check RE status
show chassis routing-engine

# Test switchover
request chassis routing-engine master switch

# Monitor during switchover
show log messages | match "GRES|Switchover"
```

NSR - Nonstop Active Routing

While GRES preserves forwarding, routing protocols still restart. NSR keeps routing protocols running during switchover.

GRES vs GRES+NSR:

```
GRES Only:                       GRES + NSR:
Forwarding: Preserved ✓          Forwarding: Preserved ✓
Protocols: Restart ✗             Protocols: Preserved ✓
Sessions: Reset ✗                Sessions: Maintained ✓
```

NSR Protocol Support

NSR replicates protocol state between REs:

Supported Protocols:

- BGP (full state)
- OSPF (full state)
- IS-IS (full state)
- LDP (label distribution)
- RSVP (MPLS TE)
- BFD sessions
- VRRP state

Junos CLI Pattern #20: NSR Configuration

```
# Enable NSR globally
set routing-options nonstop-routing

# Enable commit synchronize (required)
set system commit synchronize

# That's the basic config!
# NSR automatically works for supported protocols

# Verify NSR status
show task replication

# Output shows replicated protocols:
     Protocol        Synchronization Status
     BGP             Complete
     OSPF            Complete
     BFD             Complete
```

NSR Operational Considerations

What Gets Replicated:

```
# View replication details
show task replication detail

# Per-protocol verification
show bgp replication
show ospf replication
show bfd replication
```

What Doesn't Get Replicated:

- CLI sessions
- SNMP counters
- Traceoptions/debug
- Uncommitted configuration

Unified ISSU - In-Service Software Upgrade

ISSU allows Junos upgrades without service interruption. It combines GRES and NSR for hitless upgrades.

Traditional Upgrade:

```
1. Install new software → 2. Reboot → 3. All services down → 4. Restart
                                              ↓
                                     5-30 minutes downtime
```

With ISSU:

```
1. Upgrade backup RE → 2. Switchover → 3. Upgrade former master → 4. Done
                                  ↓
                         No service impact
```

ISSU Prerequisites

Requirements:

- Dual REs
- GRES enabled
- NSR enabled
- ISSU-compatible versions
- Sufficient resources

```
# Check ISSU compatibility
show system software validate-compatibility

# Verify prerequisites
show system switchover
show task replication
show chassis routing-engine
```

Junos CLI Pattern #21: ISSU Procedure

```
# Step 1: Verify readiness
show system switchover
show task replication

# Step 2: Stage software on backup RE
request system software add /var/tmp/junos-install-media.tgz re1

# Step 3: Validate compatibility
request system software validate /var/tmp/junos-install-media.tgz

# Step 4: Perform ISSU
request system software in-service-upgrade /var/tmp/junos-install-media.tgz

# Step 5: Monitor progress
show system in-service-upgrade status
```

ISSU Process Flow

```
# What happens during ISSU:
1. Backup RE upgrades and reboots
2. State replication resumes
```

```
3. Manual/auto switchover to upgraded RE
4. Former master upgrades
5. State replication resumes
6. Optional: Switch back to original master
```

Monitoring During ISSU

```
# Real-time ISSU monitoring
monitor start issu

# Check protocols during upgrade
show bgp summary
show ospf neighbor
show interfaces terse

# Verify forwarding continues
show pfe statistics traffic
ping 8.8.8.8 rapid count 1000
```

Troubleshooting Pattern #10: GRES/NSR/ISSU Issues

Problem 1: GRES Not Ready

```
# Check status
show system switchover

# Common issues:
# 1. Kernel database not ready
show system switchover kernel-replication

# 2. Configuration out of sync
show system commit synchronize-status

# Fix:
commit synchronize force
request system configuration rescue save
```

Problem 2: NSR Protocols Not Replicating

```
# Check replication status
show task replication detail

# Check memory usage
show task memory detail | match replicate

# Common fix:
restart routing-replication

# Or disable/re-enable
delete routing-options nonstop-routing
commit
set routing-options nonstop-routing
commit
```

Problem 3: ISSU Validation Fails

```
# Check specific failure
show log issu

# Common issues:
# 1. Version incompatibility
show version

# 2. Resource constraints
show system resource-monitor summary
```

```
# 3. Feature incompatibility
show configuration | display set | match "unsupported-feature"
```

Advanced HA Configuration

Complete HA Setup:

```
# Chassis redundancy
set chassis redundancy graceful-switchover
set chassis redundancy failover on-loss-of-keepalives
set chassis redundancy failover on-disk-failure

# Routing redundancy
set routing-options nonstop-routing

# System settings
set system commit synchronize
set system commit synchronize-server 192.168.1.1

# Interface redundancy
set interfaces ge-0/0/0 redundancy-parent ge-1/0/0
set interfaces ge-1/0/0 redundancy-parent ge-0/0/0

# High availability protocols
set protocols layer2-control nonstop-bridging
```

Real-World HA Deployment

Service Provider Core Router:

```
# Full redundancy configuration
# GRES
set chassis redundancy graceful-switchover
set chassis redundancy routing-engine 0 master
set chassis redundancy routing-engine 1 backup

# NSR
set routing-options nonstop-routing

# Commit synchronization
set system commit synchronize
set system commit synchronize force

# BFD with NSR
set protocols bfd-liveness-detection bfd-template NSR-BFD minimum-interval 300
set protocols bfd-liveness-detection bfd-template NSR-BFD multiplier 3
set protocols bfd-liveness-detection bfd-template NSR-BFD no-bfd-on-re-switchover

# BGP with NSR
set protocols bgp graceful-restart
set protocols bgp group CORE type internal
set protocols bgp group CORE bfd-liveness-detection bfd-template NSR-BFD

# OSPF with NSR
set protocols ospf graceful-restart
set protocols ospf area 0.0.0.0 interface all bfd-liveness-detection bfd-template NSR-BFD

# Monitoring
set event-options policy RE-SWITCHOVER events SYSTEM
set event-options policy RE-SWITCHOVER attributes-match system.re-switchover
set event-options policy RE-SWITCHOVER then execute-commands commands "show bgp summary"
set event-options policy RE-SWITCHOVER then execute-commands commands "show ospf neighbor"
```

Performance Impact

Resource Considerations:

```
Feature     CPU Impact    Memory Impact    Notes
------------------------------------------------
GRES        Low          Medium           Kernel sync
NSR         Medium       High             Protocol state
ISSU        High         High             During upgrade only
```

Best Practices:

```
# Monitor resources
show system resource-monitor summary
show task memory summary

# Set thresholds
set system resource-monitor resource-category kernel-memory rising-threshold 80
set system resource-monitor resource-category kernel-memory falling-threshold 70
```

HA Testing Procedures

Regular Testing Schedule:

```
# Monthly: Test GRES switchover
request chassis routing-engine master switch
show log messages | match switchover

# Quarterly: Test protocols after switchover
show bgp summary
show ospf neighbor
show interfaces terse

# Annually: Test full ISSU procedure
# (in maintenance window)
```

Building Platform HA Instincts

Health Checks:

```
# Quick HA status
show system switchover | match "On|Ready"
show task replication | match "Complete|Error"
show chassis routing-engine | match "Master|Backup"

# Detailed checks
show system switchover
show task replication detail
show chassis environment routing-engine
```

Common Mistakes:

1. Not enabling commit synchronize
2. Mismatched software versions
3. Insufficient memory for NSR
4. Not testing switchover regularly
5. Forgetting BFD no-switchover config

Operational Tips:

- Always verify before maintenance
- Document switchover procedures
- Monitor resource usage
- Test in lab first
- Have rollback plan

Integration with Previous Modules

Complete HA Stack Example:

```
# Platform HA
set chassis redundancy graceful-switchover
set routing-options nonstop-routing

# Protocol HA
set protocols ospf graceful-restart
set protocols bgp graceful-restart

# Fast detection
set protocols ospf area 0.0.0.0 interface all bfd-liveness-detection minimum-interval 300

# Gateway redundancy
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/24 vrrp-group 1 virtual-address 192.168.1.1

# Result: Complete redundancy at all layers
```

Module 15 Preview: IP Tunneling

Next, we'll explore IP tunneling technologies - GRE, IP-IP, and others. These allow you to extend networks across intermediate infrastructure and create overlay networks.

Module 15: IP Tunneling

What is IP Tunneling? - Networks Within Networks

IP tunneling is like putting a letter inside another envelope. The inner letter (original packet) travels through a postal system (intermediate network) that doesn't need to understand its contents.

Real-World Analogy:

```
Sending a package via courier:
[Your Package] → [Courier Box] → [Courier Network] → [Destination]
                     ↑                                   ↓
                  Outer wrapper                      Remove wrapper
```

In Networking Terms:

```
[Original Packet] → [Tunnel Header] → [Transport Network] → [Decapsulation]
    Payload           Encapsulation                          Original Packet
```

Why Use Tunnels?

Common Use Cases:

1. **Connect Islands**

```
Site A ←→ [Internet/MPLS] ←→ Site B
       GRE/IPsec Tunnel
```

2. **Protocol Transport**

```
IPv6 Networks ←→ [IPv4 Only Core] ←→ IPv6 Networks
                 IPv6 over IPv4
```

3. **Overlay Networks**

```
Virtual Network 1 ←→ [Physical Infrastructure] ←→ Virtual Network 1
                          Isolated via tunnels
```

4. **Security/Encryption**

```
Cleartext ←→ [IPsec Tunnel] ←→ Cleartext
           Encrypted transport
```

Types of Tunnels

1. GRE (Generic Routing Encapsulation)

- Most flexible
- Supports any protocol
- Adds GRE header
- No encryption

2. IP-IP

- Simplest tunnel
- IP packets in IP packets
- Lowest overhead
- IPv4 or IPv6

3. IPsec

- Encrypted tunnels
- Secure but complex
- Higher overhead

4. 6to4/6in4

- IPv6 over IPv4
- Transition mechanism

GRE Tunnel Fundamentals

GRE wraps packets with a new IP header and GRE header:

```
Original Packet:
[IP Header][TCP/UDP][Data]

After GRE Encapsulation:
[New IP Header][GRE Header][Original IP Header][TCP/UDP][Data]
     ↑             ↑              ↑
  Tunnel Src/Dst  Protocol ID   Preserved
```

Junos CLI Pattern #22: Basic GRE Tunnel

```
# Step 1: Create tunnel interface
set interfaces gr-0/0/0 unit 0 tunnel source 192.168.1.1
set interfaces gr-0/0/0 unit 0 tunnel destination 192.168.2.1
set interfaces gr-0/0/0 unit 0 family inet address 10.0.0.1/30

# Pattern breakdown:
# - gr-0/0/0: GRE interface (gr = GRE)
# - tunnel source: Local endpoint
# - tunnel destination: Remote endpoint
# - family inet address: Tunnel's internal IP

# Step 2: Route traffic through tunnel
set routing-options static route 172.16.0.0/16 next-hop gr-0/0/0.0
```

Complete GRE Configuration Example

Site A Configuration:

```
# Physical interface (underlay)
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.1/30
```

```
# GRE tunnel interface
set interfaces gr-0/0/0 unit 0 tunnel source 192.168.1.1
set interfaces gr-0/0/0 unit 0 tunnel destination 192.168.2.1
set interfaces gr-0/0/0 unit 0 family inet address 10.0.0.1/30

# Route remote site through tunnel
set routing-options static route 172.16.2.0/24 next-hop 10.0.0.2
```

Site B Configuration:

```
# Physical interface (underlay)
set interfaces ge-0/0/0 unit 0 family inet address 192.168.2.1/30

# GRE tunnel interface (reversed endpoints)
set interfaces gr-0/0/0 unit 0 tunnel source 192.168.2.1
set interfaces gr-0/0/0 unit 0 tunnel destination 192.168.1.1
set interfaces gr-0/0/0 unit 0 family inet address 10.0.0.2/30

# Route remote site through tunnel
set routing-options static route 172.16.1.0/24 next-hop 10.0.0.1
```

IP-IP Tunnels

IP-IP is simpler than GRE but less flexible:

```
# IP-IP tunnel configuration
set interfaces ip-0/0/0 unit 0 tunnel source 192.168.1.1
set interfaces ip-0/0/0 unit 0 tunnel destination 192.168.2.1
set interfaces ip-0/0/0 unit 0 family inet address 10.0.0.1/30

# Differences from GRE:
# - Uses ip-0/0/0 instead of gr-0/0/0
# - Lower overhead (no GRE header)
# - Only carries IP traffic
```

Advanced GRE Features

1. GRE with Key (for multiple tunnels):

```
# Add key to differentiate tunnels
set interfaces gr-0/0/0 unit 0 tunnel key 100
set interfaces gr-0/0/1 unit 0 tunnel key 200

# Both tunnels can use same source/destination
# Key differentiates traffic
```

2. GRE Keepalives:

```
# Enable keepalives for tunnel monitoring
set interfaces gr-0/0/0 unit 0 keepalive interval 10
set interfaces gr-0/0/0 unit 0 keepalive retries 3

# Tunnel goes down if keepalives fail
```

3. GRE with Fragmentation Control:

```
# Clear Don't Fragment bit
set interfaces gr-0/0/0 unit 0 clear-dont-fragment-bit

# Set MTU to avoid fragmentation
set interfaces gr-0/0/0 unit 0 family inet mtu 1476
# 1500 - 20 (IP) - 4 (GRE) = 1476
```

Running Routing Protocols Over Tunnels

OSPF over GRE:

```
# Enable OSPF on tunnel interface
set protocols ospf area 0.0.0.0 interface gr-0/0/0.0 interface-type p2p

# Benefits:
# - Dynamic routing between sites
# - Automatic failover with multiple tunnels
# - No static route maintenance
```

BGP over GRE:

```
# eBGP between sites
set protocols bgp group TUNNEL type external
set protocols bgp group TUNNEL peer-as 65002
set protocols bgp group TUNNEL neighbor 10.0.0.2
```

Tunnel Monitoring and Troubleshooting

Key Commands:

```
# Check tunnel status
show interfaces gr-0/0/0
show interfaces gr-0/0/0 extensive

# Monitor tunnel traffic
monitor traffic interface gr-0/0/0

# Check tunnel statistics
show interfaces gr-0/0/0 statistics

# Verify routing
show route table inet.0 10.0.0.0/30
```

Troubleshooting Pattern #11: Tunnel Issues

Problem 1: Tunnel Down

```
# Check 1: Can reach tunnel destination?
ping 192.168.2.1 source 192.168.1.1

# Check 2: Tunnel interface status
show interfaces gr-0/0/0 terse

# Check 3: Configuration
show configuration interfaces gr-0/0/0

# Common issues:
# - Firewall blocking GRE (Protocol 47)
# - MTU issues
# - Routing to tunnel endpoints
```

Problem 2: Traffic Not Passing Through Tunnel

```
# Check 1: Routes pointing to tunnel?
show route 172.16.2.0/24

# Check 2: Return path configured?
# (Check remote site configuration)

# Check 3: Packet capture
monitor traffic interface gr-0/0/0 no-resolve
```

Problem 3: MTU/Fragmentation Issues

```
# Symptoms: Large packets fail, small work
# Test with different sizes
ping 10.0.0.2 size 1400
ping 10.0.0.2 size 1500

# Fix: Reduce tunnel MTU
set interfaces gr-0/0/0 unit 0 family inet mtu 1400

# Or: Enable fragmentation
set interfaces gr-0/0/0 unit 0 clear-dont-fragment-bit
```

IPv6 Tunneling

6to4 Automatic Tunneling:

```
# 6to4 tunnel for IPv6 over IPv4
set interfaces gr-0/0/0 unit 0 tunnel source 192.168.1.1
set interfaces gr-0/0/0 unit 0 tunnel destination 192.168.2.1
set interfaces gr-0/0/0 unit 0 family inet6 address 2001:db8::1/64
```

Manual IPv6 over IPv4:

```
# Configure both IPv4 (tunnel) and IPv6 (payload)
set interfaces gr-0/0/0 unit 0 tunnel source 203.0.113.1
set interfaces gr-0/0/0 unit 0 tunnel destination 198.51.100.1
set interfaces gr-0/0/0 unit 0 family inet address 10.0.0.1/30
set interfaces gr-0/0/0 unit 0 family inet6 address 2001:db8:1::1/64
```

Complex Tunneling Scenarios

Hub and Spoke Topology:

```
# Hub configuration (multiple tunnels)
# Tunnel to Spoke 1
set interfaces gr-0/0/0 unit 0 tunnel source 10.0.0.1
set interfaces gr-0/0/0 unit 0 tunnel destination 10.1.1.1
set interfaces gr-0/0/0 unit 0 family inet address 172.16.0.1/30

# Tunnel to Spoke 2
set interfaces gr-0/0/1 unit 0 tunnel source 10.0.0.1
set interfaces gr-0/0/1 unit 0 tunnel destination 10.2.2.1
set interfaces gr-0/0/1 unit 0 family inet address 172.16.0.5/30

# Dynamic routing
set protocols ospf area 0.0.0.0 interface gr-0/0/0.0 interface-type p2p
set protocols ospf area 0.0.0.0 interface gr-0/0/1.0 interface-type p2p
```

Tunnel with Backup Path:

```
# Primary tunnel
set interfaces gr-0/0/0 unit 0 tunnel source 192.168.1.1
set interfaces gr-0/0/0 unit 0 tunnel destination 192.168.2.1
set interfaces gr-0/0/0 unit 0 family inet address 10.0.0.1/30

# Backup tunnel (different path)
set interfaces gr-0/0/1 unit 0 tunnel source 192.168.10.1
set interfaces gr-0/0/1 unit 0 tunnel destination 192.168.20.1
set interfaces gr-0/0/1 unit 0 family inet address 10.0.1.1/30

# Use routing protocol for automatic failover
set protocols ospf area 0.0.0.0 interface gr-0/0/0.0 metric 10
set protocols ospf area 0.0.0.0 interface gr-0/0/1.0 metric 100
```

Performance Considerations

Overhead Calculation:

```
Original packet: 1500 bytes
+ Outer IP header: 20 bytes
+ GRE header: 4-8 bytes
= Total: 1524-1528 bytes

May cause fragmentation!
```

Best Practices:

```
# 1. Set appropriate MTU
set interfaces gr-0/0/0 unit 0 family inet mtu 1476

# 2. Enable path MTU discovery
set interfaces gr-0/0/0 unit 0 tunnel path-mtu-discovery

# 3. Consider MSS clamping
set firewall filter MSS-CLAMP term 1 from protocol tcp
set firewall filter MSS-CLAMP term 1 from tcp-flags "(syn & !ack)"
set firewall filter MSS-CLAMP term 1 then tcp-mss 1436
```

Security Considerations

Securing GRE Tunnels:

```
# GRE itself provides no security!
# Options:

# 1. Firewall filters
set firewall filter GRE-FILTER term ALLOW-TUNNEL from source-address 192.168.2.1
set firewall filter GRE-FILTER term ALLOW-TUNNEL from protocol gre
set firewall filter GRE-FILTER term ALLOW-TUNNEL then accept
set firewall filter GRE-FILTER term DENY-REST then discard

set interfaces ge-0/0/0 unit 0 family inet filter input GRE-FILTER

# 2. IPsec over GRE
# (Covered in security courses)

# 3. Use IP-IP with IPsec instead
```

Real-World Deployment Example

Multi-Site Enterprise with GRE Mesh:

```
# Site A - Central Hub
# Loopback for tunnel source
set interfaces lo0 unit 0 family inet address 10.255.255.1/32

# Tunnels to each site
set interfaces gr-0/0/0 unit 0 description "Tunnel to Site-B"
set interfaces gr-0/0/0 unit 0 tunnel source 10.255.255.1
set interfaces gr-0/0/0 unit 0 tunnel destination 10.255.255.2
set interfaces gr-0/0/0 unit 0 family inet address 172.31.1.1/30

set interfaces gr-0/0/1 unit 0 description "Tunnel to Site-C"
set interfaces gr-0/0/1 unit 0 tunnel source 10.255.255.1
set interfaces gr-0/0/1 unit 0 tunnel destination 10.255.255.3
set interfaces gr-0/0/1 unit 0 family inet address 172.31.2.1/30

# iBGP over tunnels
set protocols bgp group SITES type internal
set protocols bgp group SITES local-address 10.255.255.1
set protocols bgp group SITES neighbor 172.31.1.2 description "Site-B"
set protocols bgp group SITES neighbor 172.31.2.2 description "Site-C"

# BFD for fast failure detection
```

```
set protocols bgp group SITES bfd-liveness-detection minimum-interval 1000
set protocols bgp group SITES bfd-liveness-detection multiplier 3
```

Building Tunnel Instincts

Quick Checks:

```
# Is tunnel up?
show interfaces terse | match gr-

# Traffic flowing?
show interfaces gr-0/0/0 | match rate

# Any errors?
show interfaces gr-0/0/0 extensive | match error
```

Design Patterns:

1. Use loopbacks as tunnel sources
2. Always consider MTU
3. Monitor tunnel health
4. Plan for redundancy
5. Document tunnel purposes

Common Mistakes:

1. MTU mismatches
2. Forgetting return path
3. Firewall blocking GRE
4. Using same tunnel key
5. No keepalives or BFD

Tunnel Design Best Practices

1. **Addressing Scheme:**

   ```
   Tunnel IPs: 172.31.x.x/30
   Spoke 1: 172.31.1.0/30
   Spoke 2: 172.31.2.0/30
   ```

2. **Naming Convention:**

   ```
   gr-0/0/0: HQ-to-Branch1
   gr-0/0/1: HQ-to-Branch2
   ```

3. **Monitoring:**

   ```
   set interfaces gr-0/0/0 unit 0 description "Tunnel to Branch-NYC"
   set interfaces gr-0/0/0 unit 0 keepalive interval 10
   ```

Module 16 Preview: IPv6

Finally, we'll explore IPv6 - the next generation IP protocol. We'll cover addressing, routing protocols with IPv6, and transition mechanisms.

Module 16: IPv6

IPv6 - The Next Generation Protocol

IPv6 isn't just IPv4 with more addresses. It's a redesigned protocol that solves many limitations while adding new capabilities.

Why IPv6?

```
IPv4: 4.3 billion addresses (32-bit)
IPv6: 340 undecillion addresses (128-bit)
      340,282,366,920,938,463,463,374,607,431,768,211,456

That's 295 addresses for every atom on Earth!
```

IPv6 vs IPv4 - Key Differences

Fundamental Changes:

```
Feature         IPv4                    IPv6
---------------------------------------------------------
Address Size    32 bits                 128 bits
Address Format  Dotted decimal          Hexadecimal
Header          Variable (20-60 bytes)  Fixed (40 bytes)
Fragmentation   Routers & hosts         Only hosts
Broadcast       Yes                     No (multicast instead)
ARP            Yes                     No (Neighbor Discovery)
Configuration   Manual/DHCP             SLAAC/DHCPv6/Manual
```

IPv6 Address Anatomy

IPv6 addresses are 128 bits written in hexadecimal:

```
Full Format:
2001:0db8:0000:0000:0000:0000:0000:0001

Compressed Format (remove leading zeros and :: for consecutive zeros):
2001:db8::1

Structure:
|← Global Prefix →|← Subnet →|← Interface ID →|
|    48 bits      | 16 bits  |    64 bits     |
```

IPv6 Address Types

1. Unicast (One-to-One)

```
Global Unicast:  2000::/3  (Internet routable)
Link-Local:      fe80::/10 (Single link only)
Unique Local:    fc00::/7  (Private addresses)
```

2. Multicast (One-to-Many)

```
ff00::/8
Examples:
ff02::1  - All nodes on link
ff02::2  - All routers on link
ff02::5  - All OSPF routers
```

3. Anycast (One-to-Nearest)

```
Same address on multiple interfaces
Packets go to nearest instance
```

Junos CLI Pattern #23: Basic IPv6 Configuration

```
# Enable IPv6 on interface
set interfaces ge-0/0/0 unit 0 family inet6 address 2001:db8:1::1/64
```

```
# Pattern breakdown:
# - family inet6: IPv6 address family
# - address: Full or compressed format
# - /64: Standard LAN prefix length

# Link-local automatically generated, or set manually:
set interfaces ge-0/0/0 unit 0 family inet6 address fe80::1/64
```

IPv6 Neighbor Discovery

Replaces ARP with ICMPv6-based protocol:

Key Functions:

1. **Address Resolution** (like ARP)
2. **Router Discovery**
3. **Prefix Discovery**
4. **Duplicate Address Detection**

```
# View IPv6 neighbors (like ARP cache)
show ipv6 neighbors

# Clear neighbor cache
clear ipv6 neighbors
```

SLAAC - Stateless Address Autoconfiguration

Hosts can configure themselves without DHCP:

```
1. Host creates link-local (fe80::)
2. Sends Router Solicitation
3. Router sends Router Advertisement with prefix
4. Host creates address: Prefix + Interface ID
```

Router Configuration for SLAAC:

```
# Enable router advertisements
set protocols router-advertisement interface ge-0/0/0.0 prefix 2001:db8:1::/64

# Set advertisement parameters
set protocols router-advertisement interface ge-0/0/0.0 max-advertisement-interval 600
set protocols router-advertisement interface ge-0/0/0.0 min-advertisement-interval 200
```

IPv6 Static Routing

```
# IPv6 static route
set routing-options rib inet6.0 static route ::/0 next-hop 2001:db8::1

# Or use qualified next-hop
set routing-options rib inet6.0 static route 2001:db8:2::/64 qualified-next-hop fe80::2 interface ge-0/0/0.0

# Pattern notes:
# - rib inet6.0: IPv6 routing table
# - ::/0: IPv6 default route
# - Link-local next-hops need interface
```

OSPFv3 - OSPF for IPv6

OSPFv3 is redesigned for IPv6:

```
# Basic OSPFv3 configuration
set protocols ospf3 area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf3 area 0.0.0.0 interface lo0.0 passive

# Key differences from OSPFv2:
```

```
# - Uses link-local addresses for neighbors
# - Router ID still 32-bit (usually IPv4)
# - Runs directly over IPv6
```

Complete OSPFv3 Example:

```
# Set router ID (still uses 32-bit)
set routing-options router-id 10.0.0.1

# Configure OSPFv3
set protocols ospf3 area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf3 area 0.0.0.0 interface ge-0/0/1.0
set protocols ospf3 area 0.0.0.0 interface lo0.0 passive

# Verification
show ospf3 neighbor
show ospf3 interface
show route protocol ospf3
```

BGP for IPv6

BGP easily supports IPv6 with address families:

```
# IPv6 BGP configuration
set protocols bgp group EXTERNAL type external
set protocols bgp group EXTERNAL peer-as 65002
set protocols bgp group EXTERNAL neighbor 2001:db8:1::2

# Enable IPv6 address family
set protocols bgp group EXTERNAL family inet6 unicast

# Can run IPv4 and IPv6 simultaneously
set protocols bgp group EXTERNAL neighbor 192.168.1.1
set protocols bgp group EXTERNAL family inet unicast
set protocols bgp group EXTERNAL family inet6 unicast
```

IS-IS for IPv6

IS-IS naturally supports IPv6 (protocol-independent):

```
# IPv6 with IS-IS (same as IPv4!)
set interfaces ge-0/0/0 unit 0 family inet6 address 2001:db8:1::1/64
set interfaces lo0 unit 0 family inet6 address 2001:db8:ffff::1/128

# IS-IS configuration unchanged
set protocols isis interface ge-0/0/0.0
set protocols isis interface lo0.0

# IS-IS automatically carries IPv6 routes
```

IPv6 Transition Mechanisms

1. Dual Stack (Preferred)

```
# Run IPv4 and IPv6 together
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.1/24
set interfaces ge-0/0/0 unit 0 family inet6 address 2001:db8:1::1/64

# Both protocols active simultaneously
```

2. Tunneling IPv6 over IPv4

```
# Manual tunnel (6in4)
set interfaces gr-0/0/0 unit 0 tunnel source 192.168.1.1
set interfaces gr-0/0/0 unit 0 tunnel destination 192.168.2.1
set interfaces gr-0/0/0 unit 0 family inet6 address 2001:db8:100::1/64
```

```
# Static route through tunnel
set routing-options rib inet6.0 static route 2001:db8:200::/64 next-hop gr-0/0/0.0
```

3. 6to4 Automatic Tunneling

```
# 6to4 uses 2002::/16 prefix
# IPv4 203.0.113.1 = cb00:7101 in hex
# 6to4 prefix: 2002:cb00:7101::/48

set interfaces gr-0/0/0 unit 0 tunnel source 203.0.113.1
set interfaces gr-0/0/0 unit 0 family inet6 address 2002:cb00:7101::1/64
```

IPv6 Security Features

Built-in IPsec Support:

```
# IPv6 includes IPsec headers natively
# Configure IPsec for IPv6
set security ipsec vpn V6VPN ike gateway GW1
set security ipsec vpn V6VPN establish-tunnels immediately
```

IPv6 Firewall Filters:

```
# IPv6 filter syntax
set firewall family inet6 filter V6-FILTER term ALLOW-SSH from source-address 2001:db8::/32
set firewall family inet6 filter V6-FILTER term ALLOW-SSH from next-header tcp
set firewall family inet6 filter V6-FILTER term ALLOW-SSH from destination-port 22
set firewall family inet6 filter V6-FILTER term ALLOW-SSH then accept

# Apply to interface
set interfaces ge-0/0/0 unit 0 family inet6 filter input V6-FILTER
```

IPv6 Troubleshooting

Key Commands:

```
# IPv6 connectivity
ping6 2001:db8::1
ping 2001:db8::1 source 2001:db8:1::1

# Path discovery
traceroute6 2001:db8::1

# Neighbor discovery
show ipv6 neighbors
monitor traffic interface ge-0/0/0.0 matching icmp6

# Routing
show route table inet6.0
show route protocol ospf3
```

Troubleshooting Pattern #12: IPv6 Issues

Problem 1: No IPv6 Connectivity

```
# Check 1: Interface configuration
show interfaces ge-0/0/0.0 family inet6

# Check 2: IPv6 neighbors
show ipv6 neighbors

# Check 3: Link-local connectivity first
ping fe80::2 interface ge-0/0/0.0

# Common issue: Firewall blocking ICMPv6
# ICMPv6 is mandatory for IPv6!
```

Problem 2: OSPFv3 Not Forming Adjacencies

```
# Check 1: IPv6 enabled on interface?
show ospf3 interface

# Check 2: Can ping link-local?
show ipv6 neighbors
ping fe80::2 interface ge-0/0/0.0

# Check 3: Router ID configured?
show ospf3 overview
```

Problem 3: Dual Stack Issues

```
# Verify both families configured
show interfaces ge-0/0/0.0

# Check routing tables
show route table inet.0
show route table inet6.0

# Common issue: IPv4 works, IPv6 doesn't
# Usually firewall or routing issue
```

IPv6 Best Practices

1. Address Planning:

```
/48 for sites
/64 for subnets (required for SLAAC)
/128 for loopbacks

Example hierarchy:
2001:db8::/32      - Your allocation
2001:db8:1::/48    - Site 1
2001:db8:1:1::/64  - VLAN 1
2001:db8:1:2::/64  - VLAN 2
```

2. Security Considerations:

```
# Always filter at edge
# Block ULA from Internet
set firewall family inet6 filter EDGE term BLOCK-ULA from source-address fc00::/7
set firewall family inet6 filter EDGE term BLOCK-ULA then discard

# Rate-limit ICMPv6 (but don't block!)
set firewall policer ICMP6-LIMIT bandwidth-limit 1m
```

3. Monitoring:

```
# Regular checks
show ipv6 neighbors | count
show route table inet6.0 summary
show interfaces statistics | match inet6
```

Real-World IPv6 Deployment

Enterprise Dual-Stack Network:

```
# Core Router Configuration
# Loopback
set interfaces lo0 unit 0 family inet address 10.0.0.1/32
set interfaces lo0 unit 0 family inet6 address 2001:db8:ffff::1/128

# LAN Interface
```

```
set interfaces ge-0/0/0 unit 0 family inet address 10.1.1.1/24
set interfaces ge-0/0/0 unit 0 family inet6 address 2001:db8:1:1::1/64

# WAN Interface
set interfaces ge-0/0/1 unit 0 family inet address 203.0.113.1/30
set interfaces ge-0/0/1 unit 0 family inet6 address 2001:db8:100::1/64

# OSPFv2 for IPv4
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive

# OSPFv3 for IPv6
set protocols ospf3 area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf3 area 0.0.0.0 interface lo0.0 passive

# BGP for both families
set protocols bgp group ISP type external
set protocols bgp group ISP peer-as 65100
set protocols bgp group ISP neighbor 203.0.113.2 family inet unicast
set protocols bgp group ISP neighbor 2001:db8:100::2 family inet6 unicast

# Router advertisements for LAN
set protocols router-advertisement interface ge-0/0/0.0 prefix 2001:db8:1:1::/64
set protocols router-advertisement interface ge-0/0/0.0 other-stateful-configuration
```

IPv6 Future Considerations

Emerging Trends:

1. IPv6-only networks
2. Segment Routing v6 (SRv6)
3. IPv6 in IoT
4. Happy Eyeballs (RFC 8305)

Building IPv6 Instincts

Quick Checks:

```
# IPv6 enabled?
show interfaces terse | match inet6

# Neighbors learned?
show ipv6 neighbors

# Routes present?
show route table inet6.0 summary
```

Common Mistakes:

1. Forgetting ICMPv6 is mandatory
2. Using /126 or /127 (use /64)
3. Blocking all ICMPv6
4. Not planning addresses hierarchically
5. Ignoring link-local addresses

Migration Strategy:

1. Enable dual-stack everywhere
2. Test IPv6 thoroughly
3. Monitor both protocols
4. Plan for IPv6-only future
5. Train staff on IPv6

Course Wrap-Up: Your Journey to JNCIE-SP

What You've Mastered

Through these 16 modules, you've built a complete understanding of:

Routing Fundamentals:

- How routers make decisions
- Static and dynamic routing
- The routing table structure

Interior Gateway Protocols:

- OSPF areas and optimization
- IS-IS levels and deployment
- Fast convergence techniques

BGP and Policy:

- eBGP and iBGP design
- Route reflection at scale
- Complex policy manipulation

High Availability:

- VRRP for redundancy
- BFD for fast detection
- Platform HA with GRES/NSR

Advanced Features:

- Multi-instance routing
- Load balancing techniques
- Tunneling technologies
- IPv6 deployment

Your Path Forward

1. Lab Practice:

- Build each topology
- Break things and fix them
- Time yourself on configurations

2. Real-World Application:

- Apply concepts gradually
- Document your designs
- Monitor and optimize

3. Continuous Learning:

- Follow RFC updates
- Engage with community
- Practice under pressure

Final Thoughts

Remember: Junos patterns are consistent. Once you understand the structure, you can configure any feature. Speed comes from pattern recognition, accuracy comes from understanding fundamentals.

You now have the foundation to pass the JNCIE-SP exam and excel as a service provider network engineer. Keep practicing, stay curious, and trust your instincts!